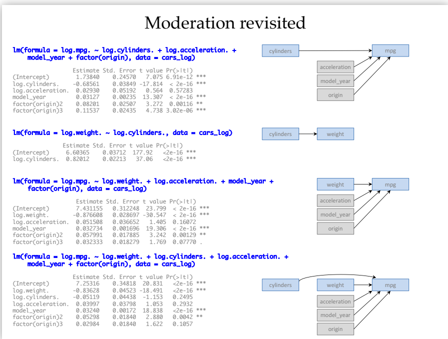
Business Analytics Using Computational Statistics

Week 12
Moderation and Mediation

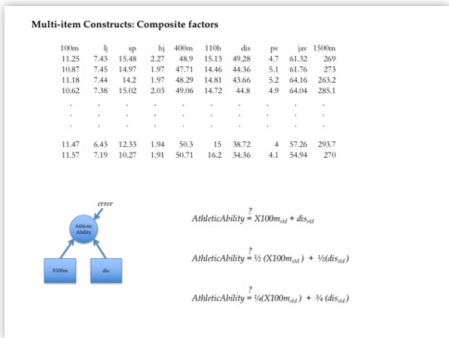
Week 13
Composites and Components

Week 14
Principal Components Analysis

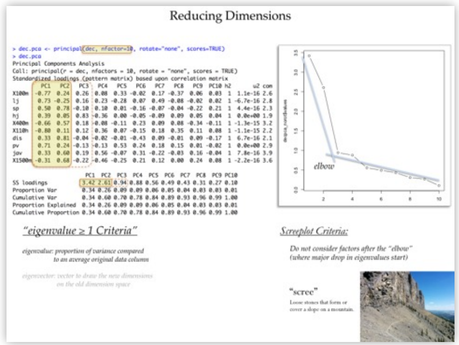
Moderation Revisited



Mediation



Principal Components



Moderation Revisited

Predicate: *asks whether a condition is true or false*

is_???: *name tells us that the results are true or false*

```
is_light_car <- cars_log$log.weight. < mean(cars_log$log.weight.)
is_heavy_car <- !is_light_car
```

We can use predicates to make our code more readable

```
light_cars_log <- cars_log[is_light_car,]
heavy_cars_log <- cars_log[is_heavy_car,]
```

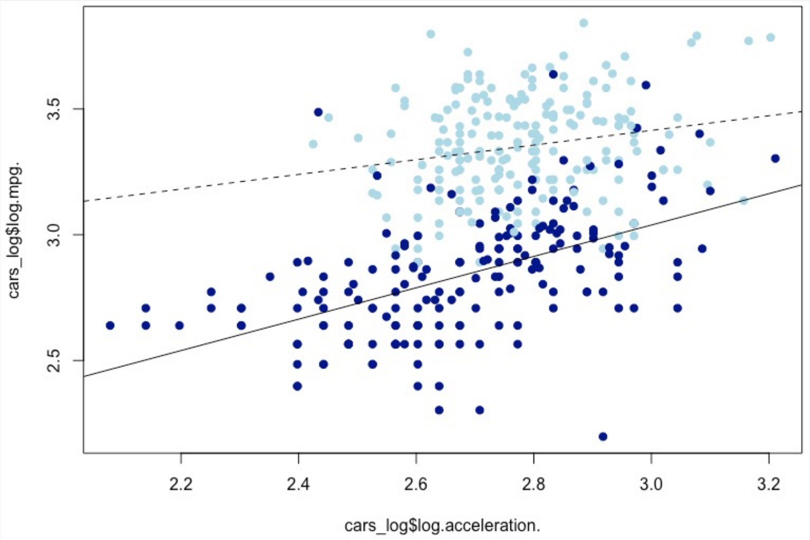
Regressions

```
lm(formula = log.mpg. ~ log.weight. + log.acceleration. +
                        model_year + factor(origin),
   data = light_cars_log)

                    Estimate Std. Error t value Pr(>|t|)
(Intercept)         6.809014   0.598446  11.378   <2e-16 ***
log.weight.        -0.821951   0.065769 -12.497   <2e-16 ***
log.acceleration.   0.111137   0.058297   1.906   0.0580 .
model_year          0.033344   0.002049  16.270   <2e-16 ***
factor(origin)2     0.042309   0.020926   2.022   0.0445 *
factor(origin)3     0.020923   0.019210   1.089   0.2774
```

```
lm(formula = log.mpg. ~ log.weight. + log.acceleration. +
                        model_year +  factor(origin),
   data = heavy_cars_log)

                    Estimate Std. Error t value Pr(>|t|)
(Intercept)         7.132892   0.677740  10.525  < 2e-16 ***
log.weight.        -0.825517   0.068101 -12.122  < 2e-16 ***
log.acceleration.   0.031221   0.055465   0.563  0.57418
model_year          0.031735   0.003254   9.752  < 2e-16 ***
factor(origin)2     0.099027   0.033840   2.926  0.00386 **
factor(origin)3     0.063148   0.065535   0.964  0.33650
```

Visualization

```
plot(cars_log$log.mpg. ~ cars_log$log.acceleration.,
     col = ifelse(is_light_car, "lightblue", "darkblue"))

abline(lm(log.mpg. ~ log.acceleration.,
       data=light_cars_log), lty="dashed")

abline(lm(log.mpg. ~ log.acceleration.,
       data=heavy_cars_log), lwd=2)
```



Slope of acceleration changes when we consider different weight categories

Statistical treatments for multicollinearity with interaction

Main terms only

```
summary(lm(log.mpg. ~ log.weight. + log.acceleration. +
                      model_year + factor(origin),
           data=cars_log))
```

```
                   Estimate  Pr(>|t|)
(Intercept)        7.431155  < 2e-16 ***
log.weight.       -0.876608  < 2e-16 ***
log.acceleration.  0.051508  0.16072
model_year         0.032734  < 2e-16 ***
factor(origin)2    0.057991  0.00129 **
factor(origin)3    0.032333  0.07770 .
```

Raw interaction

```
wt_x_acc <- with(cars_log, log.weight. * log.acceleration.)
with(cars_log, cor(log.weight., wt_x_acc))       # 0.11
with(cars_log, cor(log.acceleration., wt_x_acc)) # 0.85

summary(lm(log.mpg. ~ log.weight. + log.acceleration. + wt_x_acc +
                      model_year + factor(origin), data=cars_log))
```

High correlation between interaction term and independent variable

```
                   Estimate  Pr(>|t|)
(Intercept)        1.089642  0.69245
log.weight.       -0.096632  0.77488
log.acceleration.  2.357574  0.01834 *
wt_x_acc          -0.287170  0.02094 *
model_year         0.033685  < 2e-16 ***
factor(origin)2    0.058737  0.00105 **
factor(origin)3    0.028179  0.12370
```

Mean-Centered interaction

```
log.weight.mc <- scale(cars_log$log.weight., scale=FALSE)
log.acceleration.mc <- scale(cars_log$log.acceleration., scale=FALSE)
wt_x_acc_mc <- log.weight.mc * log.acceleration.mc

with(cars_log, cor(log.weight., wt_x_acc.mc))       # -0.20
with(cars_log, cor(log.acceleration., wt_x_acc.mc)) #  0.35
summary(lm(log.mpg. ~ log.weight. + log.acceleration. + wt_x_acc_mc +
                      model_year + factor(origin), data=cars_log))
```

```
                   Estimate  Pr(>|t|)
(Intercept)        7.325939  < 2e-16 ***
log.weight.       -0.880393  < 2e-16 ***
log.acceleration.  0.072596  0.05403 .
wt_x_acc.s        -0.287170  0.02094 *
model_year         0.033685  < 2e-16 ***
factor(origin)2    0.058737  0.00105 **
factor(origin)3    0.028179  0.12370
```

Orthogonalized interaction

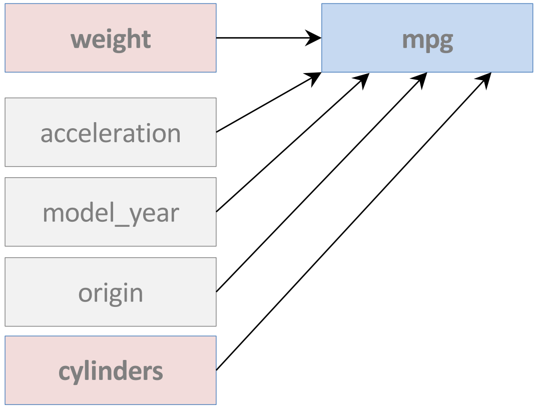
```
oregr <- with(cars_log, lm(wt_x_acc ~ log.weight. + log.acceleration.))
wt_x_acc_ortho <- oregr$residuals

summary(lm(log.mpg. ~ log.weight. + log.acceleration. + wt_x_acc_ortho +
                      model_year + factor(origin), data=cars_log))
```

```
                   Estimate  Pr(>|t|)
(Intercept)        7.377176  < 2e-16 ***
log.weight.       -0.876967  < 2e-16 ***
log.acceleration.  0.046100  0.20764
wt_acc            -0.287170  0.02094 *
model_year         0.033685  < 2e-16 ***
factor(origin)2    0.058737  0.00105 **
factor(origin)3    0.028179  0.12370
```

These statistical techniques
cannot remove the multicollinearity <u>problem</u>!

But they can give us
***more interpretable** coefficients*

Mediation Revisited



Direct Effects on Outcome

```
mpg_all <- lm(
  log.mpg. ~ log.weight. + log.cylinders. +
    log.acceleration. + model_year + factor(origin),
  data=cars_log)
summary(mpg_all)
```

```
Coefficients:
                  Estimate Std. Error t value Pr(>|t|)
(Intercept)        7.25316    0.34818  20.831   <2e-16 ***
log.weight.       -0.83628    0.04523 -18.491   <2e-16 ***
log.cylinders.    -0.05119    0.04438  -1.153   0.2495
log.acceleration.  0.03997    0.03798   1.053   0.2932
model_year         0.03240    0.00172  18.838   <2e-16 ***
factor(origin)2    0.05298    0.01840   2.880   0.0042 **
factor(origin)3    0.02984    0.01840   1.622   0.1057
```

Multicollinearity problems?

```
round(with(cars_log,
  cor(data.frame(
    mpg=log.mpg., wgt=log.weight., cyl=log.cylinders.,
    acc=log.acceleration., yr=model_year, ogn=origin))
), 2)
```

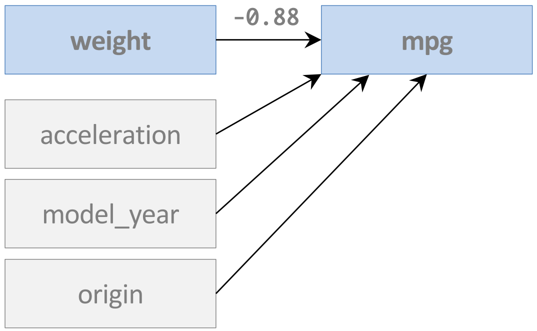
	mpg	wgt	cyl	acc	yr	ogn
mpg	1.00	-0.87	-0.82	0.46	0.58	0.56
wgt	-0.87	1.00	0.88	-0.43	-0.28	-0.60
cyl	-0.82	0.88	1.00	-0.50	-0.34	-0.58
acc	0.46	-0.43	-0.50	1.00	0.31	0.22
yr	0.58	-0.28	-0.34	0.31	1.00	0.18
ogn	0.56	-0.60	-0.58	0.22	0.18	1.00

Multicollinearity prevents us from interpreting the coefficients and their standard errors

```
car::vif(mpg_all)
                      GVIF
log.weight.       4.788498
log.cylinders.    5.321090
log.acceleration. 1.400111
model_year        1.201815
factor(origin)    1.792784
```

But what if we know that cylinders only reduces mpg because it adds weight to the car?

Bootstrapped Mediation Revisited



Direct Effects on Outcome

```r
mpg_wt <- lm(
  log.mpg. ~ log.weight. +
             log.acceleration. + model_year + factor(origin),
  data=cars_log)
```

	Estimate	Pr(>|t|)	
(Intercept)	7.431155	< 2e-16	***
log.weight.	**-0.876608**	< 2e-16	***
log.acceleration.	0.051508	0.16072	
model_year	0.032734	< 2e-16	***
factor(origin)2	0.057991	0.00129	**
factor(origin)3	0.032333	0.07770	.

Antecedent Effect on Mediator

```r
wt_cyl <- lm(log.weight. ~ log.cylinders., data=cars_log)
```

	Estimate	Pr(>|t|)	
(Intercept)	6.60365	<2e-16	***
log.cylinders.	**0.82012**	<2e-16	***

Indirect Effect

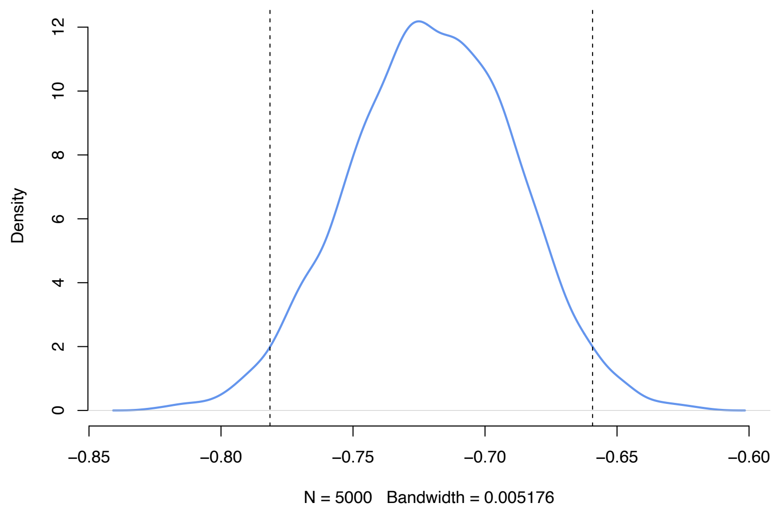
0.82 * -0.88 = **-0.72**


```r
boot_mediation <- function(model1, model2, dataset) {
  boot_index <- sample(1:nrow(dataset), replace=TRUE)
  data_boot <- dataset[boot_index, ]
  regr1 <- lm(model1, data_boot)
  regr2 <- lm(model2, data_boot)
  return( regr1$coefficients[2] * regr2$coefficients[2] )
}

indirect <- replicate(2000,
                boot_mediation(wt_cyl, mpg_wt, cars_log))

quantile(indirect, probs=c(0.025, 0.975))
```

```
      2.5%      97.5%
-0.7814449 -0.6592682
```

95% CI of Indirect effect

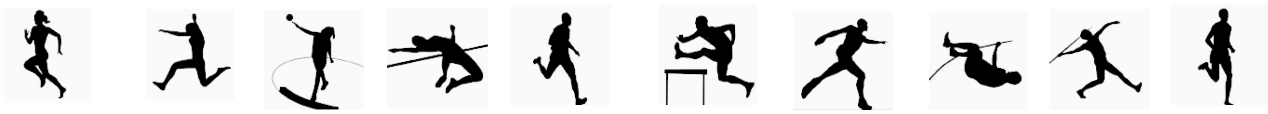
```r
plot(density(indirect), …)
abline(v=quantile(indirect, probs=c(0.025, 0.975)), …)
```



Decathlon Dataset

different units

100m	long jump	shot put	high jump	400m	110m hurdles	discus throw	pole vault	javelin throw	1500m



	X100m	lj	sp	hj	X400m	X110h	dis	pv	jav	X1500m
1	11.25	7.43	15.48	2.27	48.90	15.13	49.28	4.7	61.32	268.95
2	10.87	7.45	14.97	1.97	47.71	14.46	44.36	5.1	61.76	273.02
3	11.18	7.44	14.20	1.97	48.29	14.81	43.66	5.2	64.16	263.20
4	10.62	7.38	15.02	2.03	49.06	14.72	44.80	4.9	64.04	285.11
	.	.	.	.	.	.	.	.	.	.
	.	.	.	.	.	.	.	.	.	.
32	11.47	6.43	12.33	1.94	50.30	15.00	38.72	4.0	57.26	293.72
33	11.57	7.19	10.27	1.91	50.71	16.20	34.36	4.1	54.94	269.98

Athletic Ability

Construct: a concept we are interested in understanding

10 sports representing athletic ability were selected by experts & tradition.

How many unique dimensions of athletic ability are represented here? 🤔

```
dec <- read.table("decathlon_data.txt", header=T)
round( cor(dec), 2)
```

	X100m	lj	sp	hj	X400m	X110h	dis	pv	jav	X1500m
X100m	1.00	-0.54	-0.21	-0.15	0.61	0.64	-0.05	-0.39	-0.06	0.26
lj	-0.54	1.00	0.14	0.27	-0.52	-0.48	0.04	0.35	0.18	-0.40
sp	-0.21	0.14	1.00	0.12	0.09	-0.30	0.81	0.48	0.60	0.27
hj	-0.15	0.27	0.12	1.00	-0.09	-0.31	0.15	0.21	0.12	-0.11
X400m	0.61	-0.52	0.09	-0.09	1.00	0.55	0.14	-0.32	0.12	0.59
X110h	0.64	-0.48	-0.30	-0.31	0.55	1.00	-0.11	-0.52	-0.06	0.14
dis	-0.05	0.04	0.81	0.15	0.14	-0.11	1.00	0.34	0.44	0.40
pv	-0.39	0.35	0.48	0.21	-0.32	-0.52	0.34	1.00	0.27	-0.03
jav	-0.06	0.18	0.60	0.12	0.12	-0.06	0.44	0.27	1.00	0.10
X1500m	0.26	-0.40	0.27	-0.11	0.59	0.14	0.40	-0.03	0.10	1.00

Can you tell which dimensions of athletics are captured by decathlon sports?

Expert driven *feature selection*

Can we use algorithms to find the key dimensions in this decathlon dataset?

Data driven *feature selection*

Multi-Item Measurements

two-variable example

Before we examine the dimensionality of the 10 variable decathlon problem, let's consider an easier 2 variable problem.

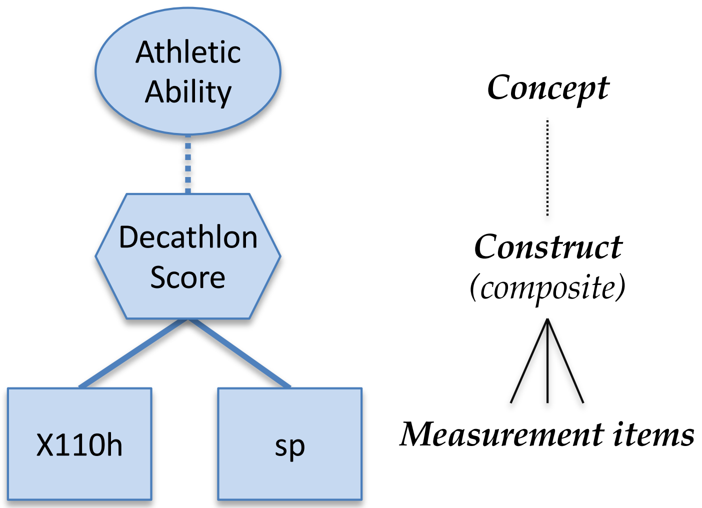
Imagine an event with just two events:
110m hurdles and **shotput**

How can we:

*(a) Meaningfully **compare performance***
*(b) Give a **single score/rank** to each athlete*

Composite Measurement

(combining measurements)



How can we weight these two items to measure athletic ability?

$$\boldsymbol{AthleticAbility = w_1 \cdot X110h + w_2 \cdot sp}$$

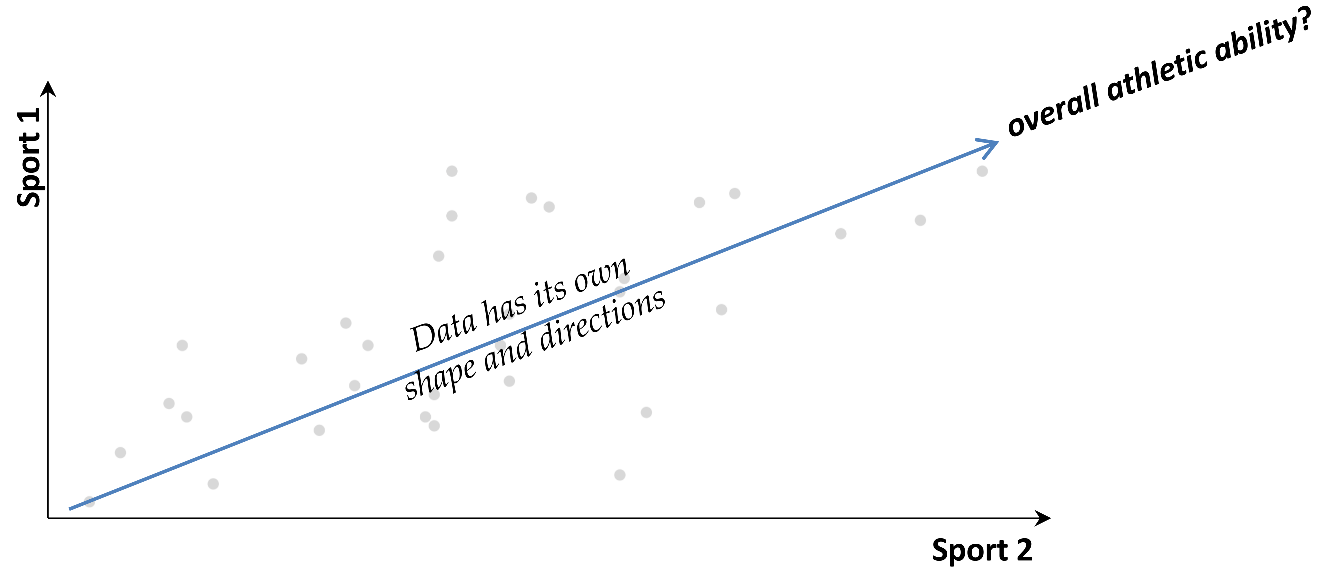
$$AthleticAbility = X110h + sp$$

$$AthleticAbility = \tfrac{1}{2}\,(X110h\,) + \tfrac{1}{2}\,(sp\,)$$

$$AthleticAbility = \tfrac{1}{4}(X110h\,) + \tfrac{3}{4}\,(sp\,)$$

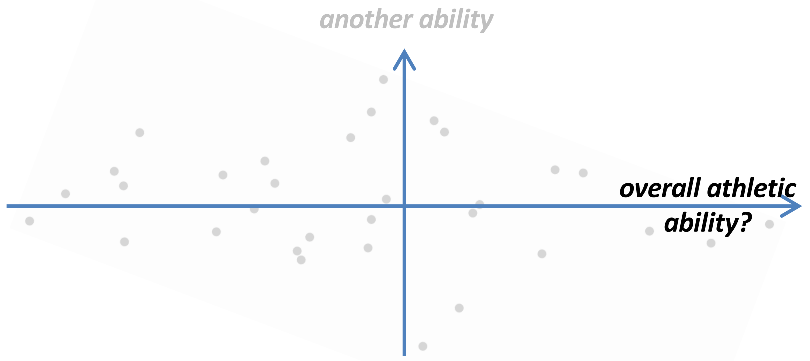
Creating Composites Geometrically

Data does not usually follow our axes of measurement…



Dimension Transformation

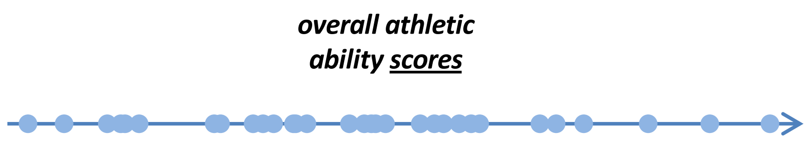
What if we could re-orient our data so that the axes were helpful for our analysis?



*We could **focus on the dimensions** we are interested in and explore what the remaining dimensions mean*

Dimension Reduction

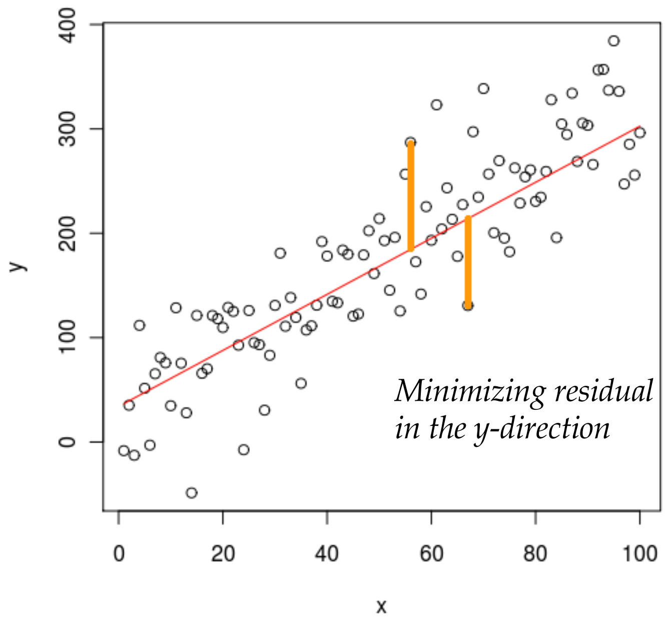
What if we could *score* each case with fewer variables than in the original dataset?



*We could **reduce our data** to a single variable for ranking etc.*

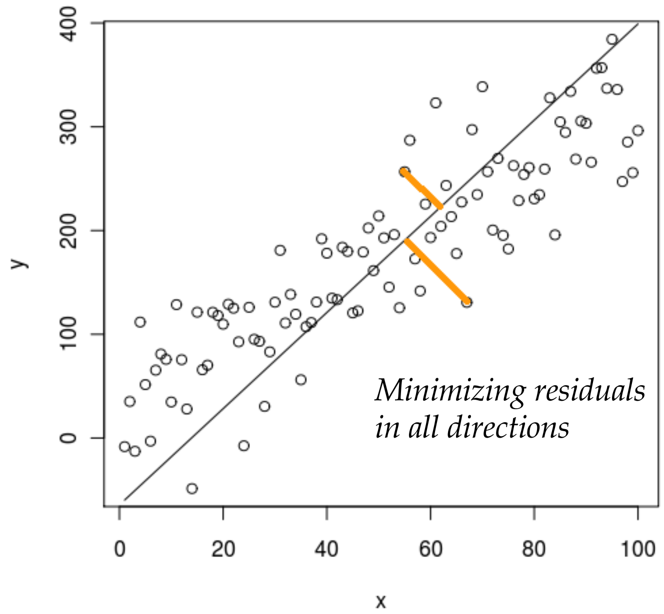
Regression vs. **Principal Components**

Regression line



versus

Principal Component



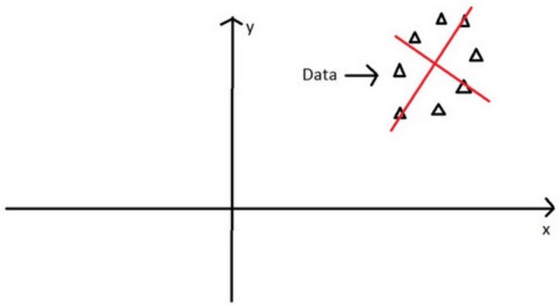
Recall that we treat the regression line as if it tries to **explain or predict** the values of **y**.

We treat the principal component as a new dimension that **summarizes** the variance of **x and y**.

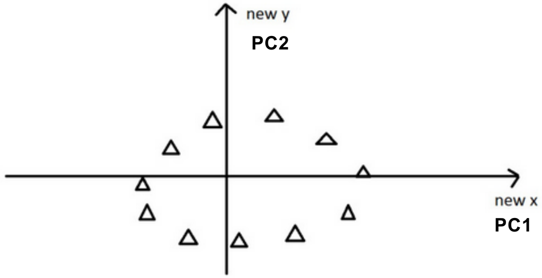
Source: http://www.cerebralmastication.com/2010/09/principal-component-analysis-pca-vs-ordinary-least-squares-ols-a-visual-explination/

Capturing Variance using Principal Components

Principal components are the directions in which a set of data points stretch to express their variance.



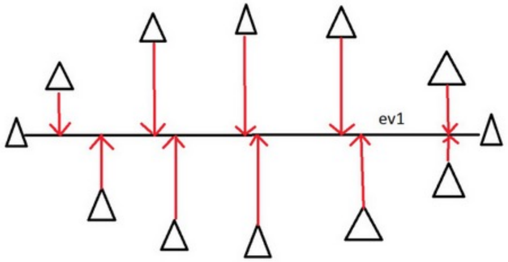
Principal components are ***orthogonal vectors*** that can be new, natural dimensions for the data



We can create as many principal components as there are original dimensions in the data

(x, y) → (PC1, PC2)

The vector that captures the most ***orthogonal*** variance is called the ***first principal component***.



The vector that best captures the remaining variance is called the ***second principal component***.

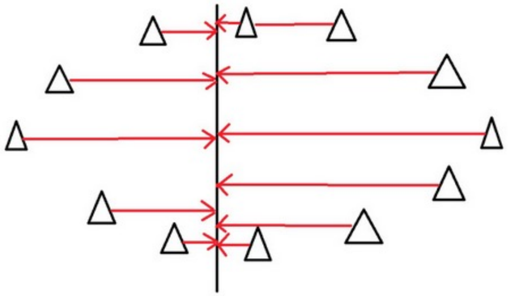
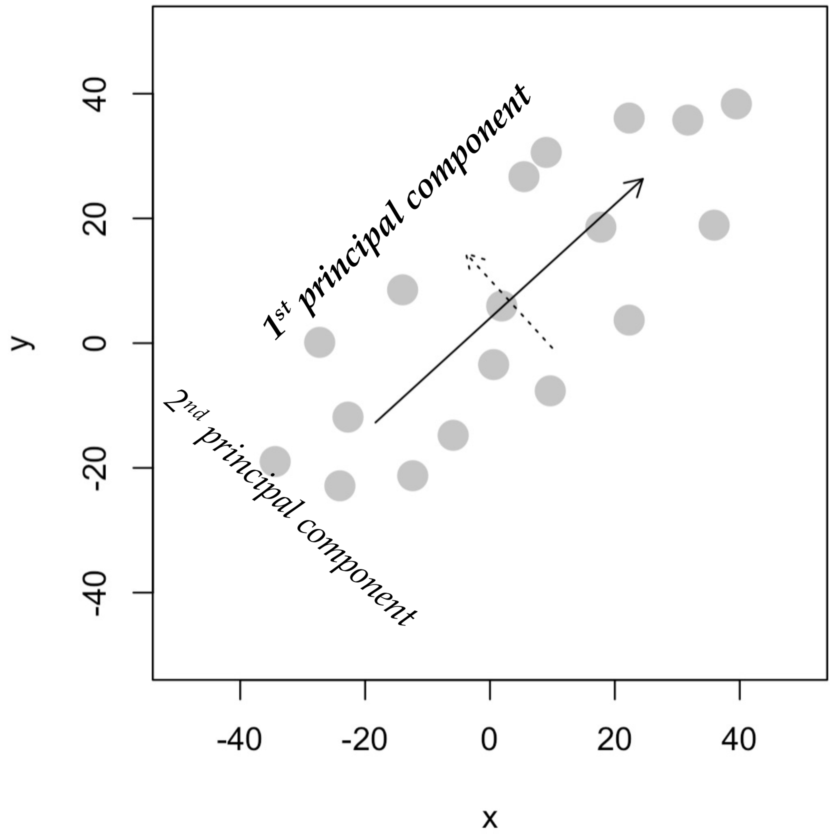


Image source: http://georgemdallas.wordpress.com/2013/10/30/principal-component-analysis-4-dummies-eigenvectors-eigenvalues-and-dimension-reduction/

PCA Interactive Demonstration




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

library(compstatslib)
interactive_pca()
```

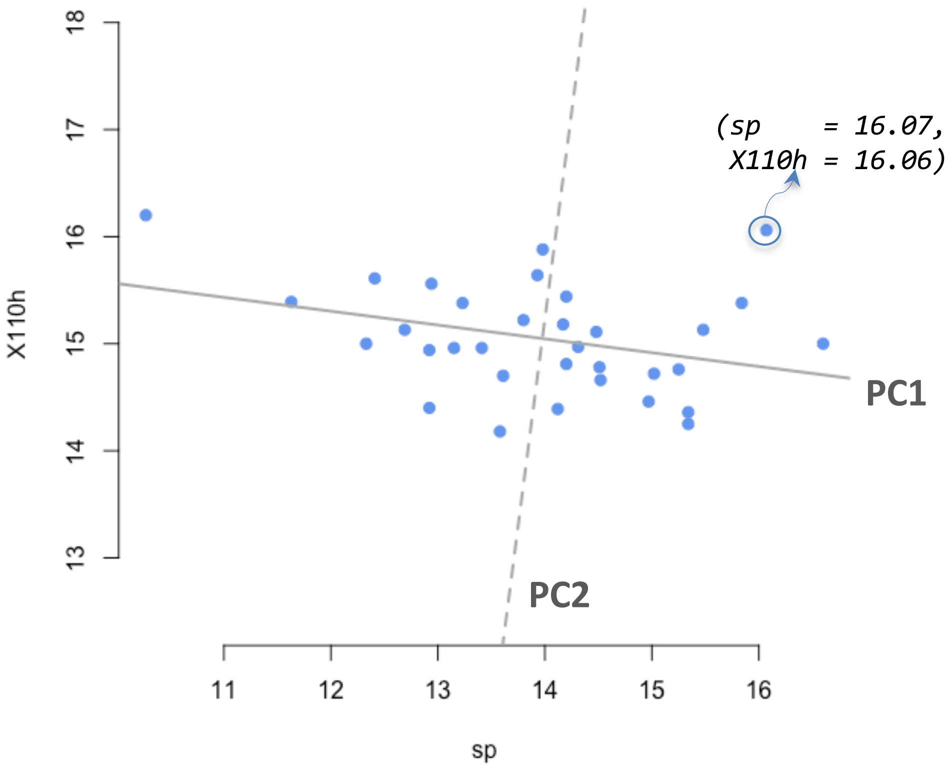
Uses of PCA

Dimension Transformation and Reduction

Let's take a simplistic 2-variable example:

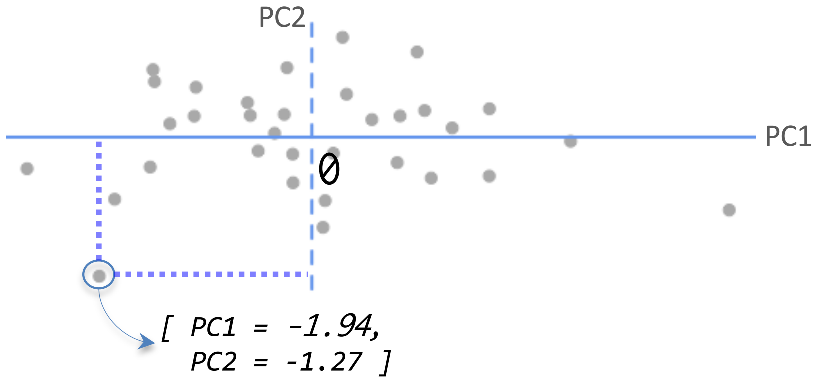
```
dec2 <- dec[,c("sp", "X110h")]
```

Original Measurements



Dimension Transformation

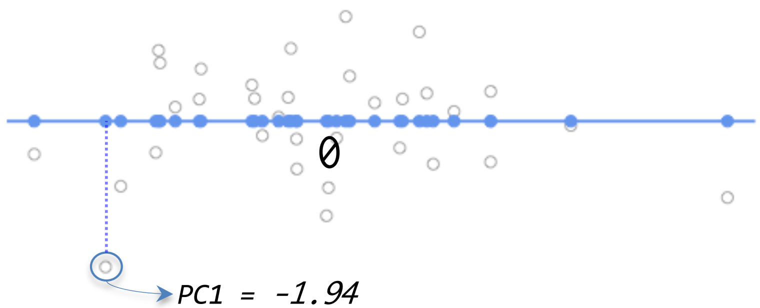
We can transform each (x, y) data point to a more orthogonal, standardized coordinate system.



*We have **scores** of PC1 and PC2 that describe each point*

Dimension Reduction

*We can summarize each combination of [sp, X110h] using a single **score** on the first principal component.*



We can reduce our two dimensional data into one dimension by just taking the PC1 score!

Principal Components as Eigenvectors

The **principal components of a dataset** are the **eigenvectors of its covariance matrix**

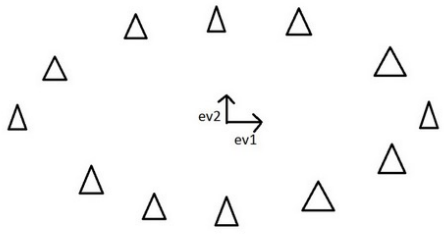
```r
cov(dec2)
```

```
          sp X110h
sp      1.77 -0.20
X110h  -0.20  0.26
```

diagonal: *variances of each variable*
off-diagonals: *dot products of the two variables*

```r
dec2_eigen <- eigen(cov(dec2))
```

`eigen()` : *compute eigenvalues and eigenvectors*




```r
dec2_eigen$vectors
```

	PC1	PC2
	[,1]	[,2]
sp [1,]	-0.9917381	-0.1282794
X110h [2,]	0.1282794	-0.9917381

Eigenvectors

Each principal component's ***direction*** of variance

Expressed in terms of x and y

Units of **`sp`** *for 1 unit of PC*

Units of **`X110h`** *for 1 unit of PC*

```r
sum(dec2_eigen$vectors[,1]^2) # 1
sum(dec2_eigen$vectors[,2]^2) # 1
```

Each eigenvector's ***magnitude*** (length) is 1

$$EV1\colon \sqrt{-0.991^2 + 0.128^2} = 1$$

$$EV2\colon \sqrt{-0.128^2 + -0.991^2} = 1$$

Eigenvalues

```r
dec2_eigen$values
```

```
[1] 1.8000181 0.2309917
```

***PC1** captures 1.80 times the **variance**…*

***PC2** captures 0.23 times the **variance**…*

…of the average of the original variables (sp and X110h)

Each principal component's ***magnitude***

***Variance captured** by PC relative to average original data dimension*

Variance of original dataset captured by PC (eigenvalues sum to number of dimensions)

Strange values? See: https://stats.stackexchange.com/questions/140434/does-a-correlation-matrix-of-two-variables-always-have-the-same-eigenvectors

prcomp() function

```
dec2_pca <- prcomp(dec2)
```

```
Standard deviations (1, .., p=2):

[1] 1.341648 0.480616

Rotation (n x k) = (2 x 2):
            PC1        PC2
sp    -0.9917381  0.1282794
X110h  0.1282794  0.9917381
```

prcomp() : *Perform principal components on data*
scale. : *standardize all data to reduce effect of different scales*

Standard deviation: *explained by principal components relative to original items (square root of eigenvalues)*

Eigenvectors:
Relationship of PCs compared to original items

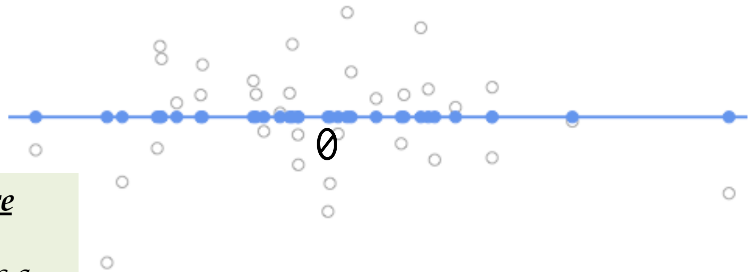
NOTE: direction of PCs can change depending on PCA package's estimation method

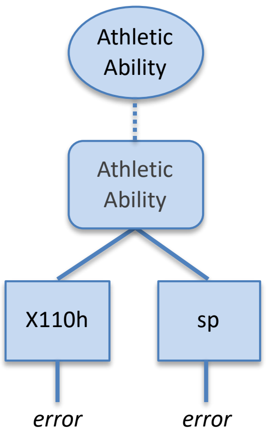
```
scores = dec2_pca$x
```

```
             PC1         PC2
 [1,] -1.4807956  0.27342666
 [2,] -1.0609563 -0.45646032
  …         …           …
[32,]  1.6265030 -0.25957927
[33,]  3.8234187  0.66625093
```

$x *report scores*

Scores*: new values for original observations based on PCs*



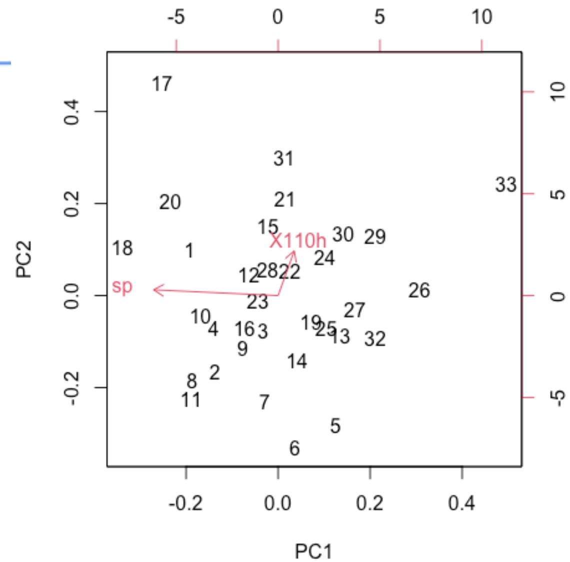


Composite Score

*We can use **PC1** as a **composite score** for X110h and sp!*

```
scores[,"PC1"]
 [1] -1.4807956 -1.0609563 -0.2524203 -1.0771906  0.9644099  0.2816414 -0.2269585
 [8] -1.4407274 -0.5890183 -1.3001592 -1.4548381 -0.4916231  1.0336808  0.3185945
[15] -0.1716043 -0.5637074 -1.9466212 -2.6082185  0.5502947 -1.8057515  0.1218209
[22]  0.1968695 -0.3409867  0.7826849  0.8081466  2.3707486  1.2861536 -0.1752047
[29]  1.6254144  1.0933792  0.1030210  1.6265030  3.8234187
```

```
biplot(dec2_pca)
```



Different Scales

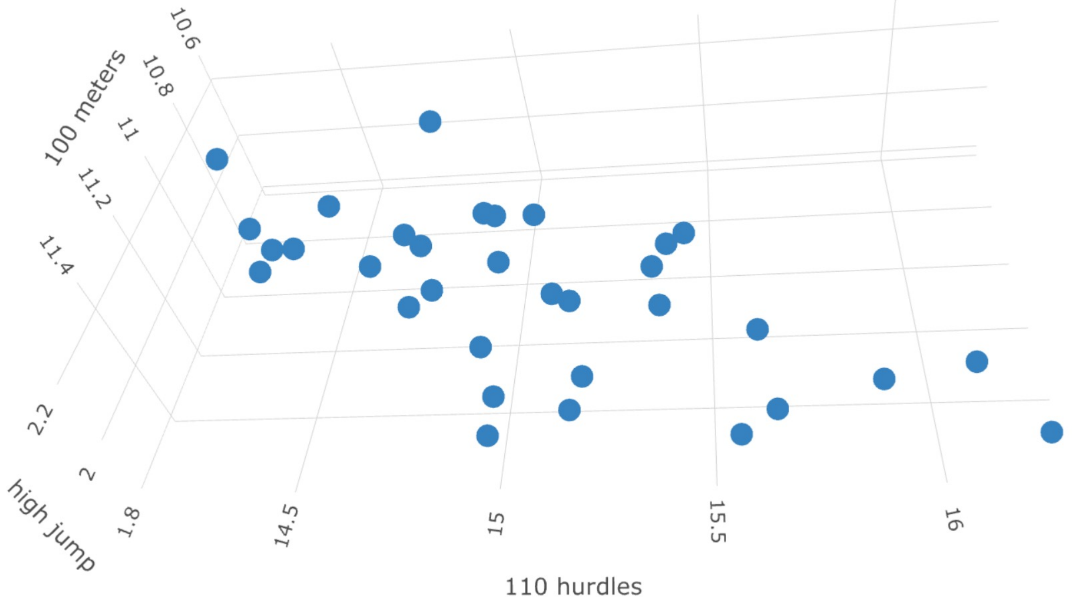
Let's do PCA with three different sports:

```
dec3 <- dec[,c("hj","X100m", "X110h")]
```

These three scales have **different range**:

```
sapply(dec3, \(x) { max(x) - min(x) })

  hj X100m X110h
0.48  0.95  2.02
```

Maximum variance (stretch) will always be in direction of 110 hurdles



PCA is heavily influenced by larger scales
We cannot analyze the covariance matrix
if our data columns do not have similar scales

```
prcomp(dec3)
 Standard deviations (1, .., p=3):
 [1] 0.53398076 0.17832703 0.08895543

 Rotation (n x k) = (3 x 3):
               PC1         PC2         PC3
 hj     0.05382043 -0.06994725 -0.99609776
 X100m -0.32923619 -0.94300481  0.04842994
 X110h -0.94271252  0.32534491 -0.07378209
```

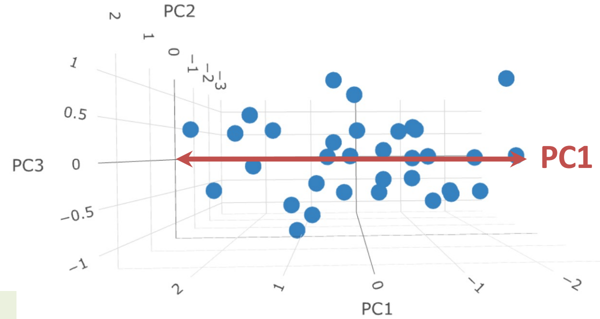
110 hurdles dominates PC1 because its scale is wider

Scaled Matrix

*The **correlation matrix** is a fully standardized covariance matrix!*

```
cor(dec3)
             hj      X100m      X110h
hj     1.0000000 -0.1459075 -0.3067350
X100m -0.1459075  1.0000000  0.6383615
X110h -0.3067350  0.6383615  1.0000000
```

100m and 110h have high correlation



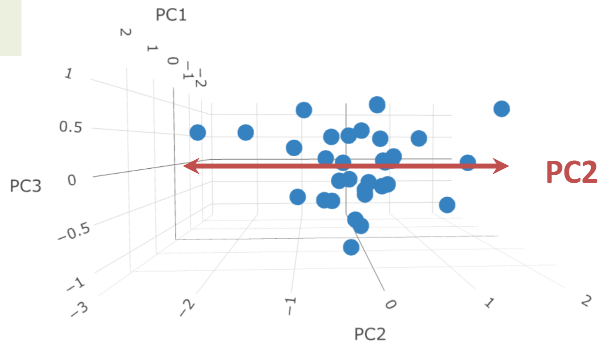
Let's analyze the correlation matrix:

```
dec3_eigen <- eigen(cor(dec3))
 eigen() decomposition
 $values
 [1] 1.7725560 0.8880372 0.3394069

 $vectors
           [,1]      [,2]       [,3]
[1,]  0.3864727 0.9021702  0.1916447
[2,] -0.6297665 0.4099401 -0.6598055
[3,] -0.6738197 0.1343054  0.7265873
```

Eigenvalues of scaled matrix sum up to number of original dimensions!

```
sum(dec3_scaled_eigen$values)
 [1] 3
```

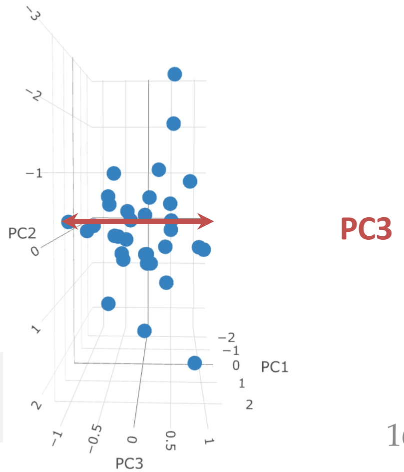
***PC1** equally qually captures 110m and 110h*
***PC2** mostly captures high jump*



Scaling the data gives us the same results:

```
dec3_pca <- prcomp(scale(dec3))
dec3_pca <- prcomp(dec3, scale. = TRUE)
 Standard deviations (1, .., p=3):
 [1] 1.3313737 0.9423572 0.5825864

 Rotation (n x k) = (3 x 3):
              PC1        PC2        PC3
 hj    -0.3864727 -0.9021702  0.1916447
 X100m  0.6297665 -0.4099401 -0.6598055
 X110h  0.6738197 -0.1343054  0.7265873
```

***PC3** doesn't capture much variance of the original data*



How much variance does each PC capture?

Principal Components Analysis

```r
dec <- read.table("decathlon_data.txt", header=T)
```

	X100m	lj	sp	hj	X400m	X110h	dis	pv	jav	X1500m
1	11.25	7.43	15.48	2.27	48.90	15.13	49.28	4.7	61.32	268.95
2	10.87	7.45	14.97	1.97	47.71	14.46	44.36	5.1	61.76	273.02
3	11.18	7.44	14.20	1.97	48.29	14.81	43.66	5.2	64.16	263.20
4	10.62	7.38	15.02	2.03	49.06	14.72	44.80	4.9	64.04	285.11
	.	.	.	.	.	.	.	.	.	.
	.	.	.	.	.	.	.	.	.	.
32	11.47	6.43	12.33	1.94	50.30	15.00	38.72	4.0	57.26	293.72
33	11.57	7.19	10.27	1.91	50.71	16.20	34.36	4.1	54.94	269.98

Athletic Ability

Construct: a concept we are interested in understanding

10 sports representing athletic ability were selected by experts & tradition.

How many unique dimensions of athletic ability are represented here?

```r
round( cor(dec), 2)
```

	X100m	lj	sp	hj	X400m	X110h	dis	pv	jav	X1500m
X100m	1.00	-0.54	-0.21	-0.15	0.61	0.64	-0.05	-0.39	-0.06	0.26
lj	-0.54	1.00	0.14	0.27	-0.52	-0.48	0.04	0.35	0.18	-0.40
sp	-0.21	0.14	1.00	0.12	0.09	-0.30	0.81	0.48	0.60	0.27
hj	-0.15	0.27	0.12	1.00	-0.09	-0.31	0.15	0.21	0.12	-0.11
X400m	0.61	-0.52	0.09	-0.09	1.00	0.55	0.14	-0.32	0.12	0.59
X110h	0.64	-0.48	-0.30	-0.31	0.55	1.00	-0.11	-0.52	-0.06	0.14
dis	-0.05	0.04	0.81	0.15	0.14	-0.11	1.00	0.34	0.44	0.40
pv	-0.39	0.35	0.48	0.21	-0.32	-0.52	0.34	1.00	0.27	-0.03
jav	-0.06	0.18	0.60	0.12	0.12	-0.06	0.44	0.27	1.00	0.10
X1500m	0.26	-0.40	0.27	-0.11	0.59	0.14	0.40	-0.03	0.10	1.00

Can we use algorithms to find the key dimensions in this decathlon dataset?

Data driven *feature selection*

Let's analyze the dimensionality using ***PCA****: Principal Components Analysis*

PCA Dimensions

10 variable example

```
dec_eigen <- eigen(cor(dec))
dec_eigen$values
  [1] 3.4182381 2.6063931 0.9432964 0.8780212 0.5566267 0.4912275 0.4305952 0.3067981 0.2669494 0.1018542
```

```
dec_pca <- prcomp(dec, scale. = TRUE)
 Standard deviations:
  [1] 1.8488478 1.6144328 0.9712345 0.9370279 0.7460742 0.7008762 0.6561975 0.5538936 0.5166715 0.3191460

 Rotation:
```

	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8	PC9	PC10
X100m	-0.4158823	0.1488081	-0.26747198	0.08833244	-0.442314456	0.03071237	0.2543985	-0.663712826	0.10839531	-0.10948045
lj	0.3940515	-0.1520815	-0.16894945	0.24424963	0.368913901	-0.09378242	0.7505343	-0.141264141	-0.04613910	-0.05580431
sp	0.2691057	0.4835374	0.09853273	0.10776276	-0.009754680	0.23002054	-0.1106637	-0.072505560	-0.42247611	-0.65073655
hj	0.2122818	0.0278985	-0.85498656	-0.38794393	-0.001876311	0.07454380	-0.1351242	0.155435871	0.10206505	-0.11941181
X400m	-0.3558474	0.3521598	-0.18949642	-0.08057457	0.146965351	-0.32692886	0.1413388	0.146839303	-0.65076229	0.33681395
X110h	-0.4334816	0.0695682	-0.12616012	0.38229029	-0.088802794	0.21049130	0.2725296	0.639003579	0.20723854	-0.25971800
dis	0.1757923	0.5033347	0.04609969	-0.02558404	0.019358607	0.61491241	0.1439726	-0.009400445	0.16724055	0.53450315
pv	0.3840821	0.1495820	0.13687235	-0.14396548	-0.716743474	-0.34776037	0.2732665	0.276873049	0.01766443	0.06589572
jav	0.1799436	0.3719570	-0.19232803	0.60046566	0.095582043	-0.43744387	-0.3419099	-0.058519366	0.30619617	0.13093187
X1500m	-0.1701426	0.4209653	0.22255233	-0.48564231	0.339772188	-0.30032419	0.1868704	-0.007310045	0.45688227	-0.24311846

What might these new dimensions mean?

```
summary(dec_pca)
```

Importance of components:

	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8	PC9	PC10
Standard deviation	1.8488	1.6144	0.97123	0.9370	0.74607	0.70088	0.65620	0.55389	0.51667	0.31915
Proportion of Variance	**0.3418**	**0.2606**	**0.09433**	**0.0878**	**0.05566**	**0.04912**	**0.04306**	**0.03068**	**0.02669**	**0.01019**
Cumulative Proportion	0.3418	0.6025	0.69679	0.7846	0.84026	0.88938	0.93244	0.96312	0.98981	1.00000

How much variance of the dataset each PC is capturing

Eigenvalue / (# of dimensions)

```
dec_eigen$values[1] / sum(dec_eigen$values)
 [1] 0.3418238
```

34.18%

Reducing Dimensions

How many dimensions of principal components should we pick?

Eigenvalue ≥ 1 Criteria

Which PCs capture more variance than the original data items, on average?

Only consider PCs with eigenvalues ≥ 1

```
dec_eigen$values

$values
  [1] 3.4182381 2.6063931 0.9432964 0.8780212 …
```

Choose only first two (or three?) principal components

Screeplot Criteria:

Which PCs offer significantly more variance than the remaining PCs?

Only consider factors <u>before</u> the "elbow"

"scree" — Loose stones that fall and cover a slope on a mountain.

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