Principal Component Regression
and
Partial Least Squares (PLS)

Biased Regression Techniques

UNbiased Regression

$$\mathbf{X}\hat{\boldsymbol{\beta}} = \hat{\mathbf{y}}$$

$$E(\hat{\boldsymbol{\beta}}) = \boldsymbol{\beta}$$

mean of the distribution of all possible sample parameters

population "truth"

Biased Regression

$$\mathbf{X}\hat{\boldsymbol{\beta}} = \hat{\mathbf{y}}$$

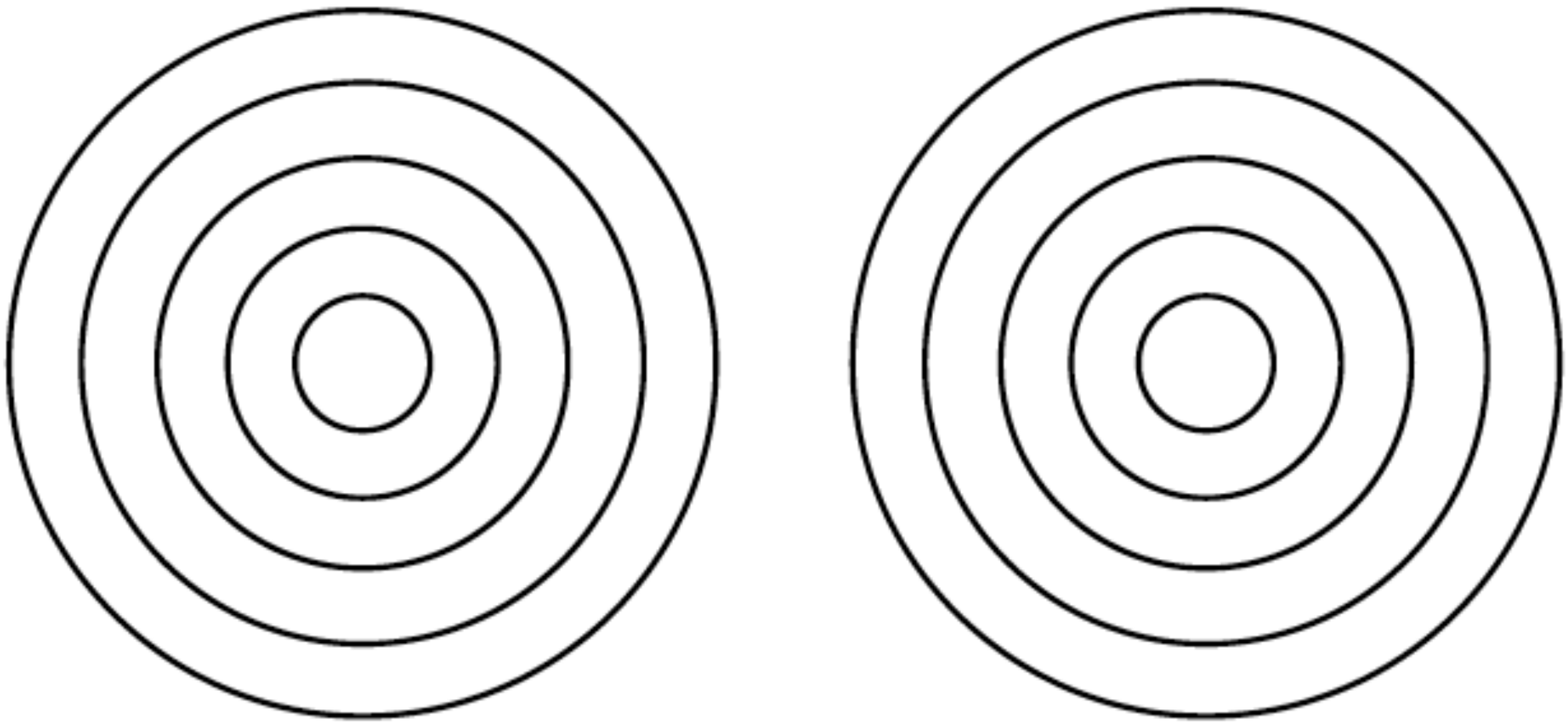
on purpose!

$$E(\hat{\boldsymbol{\beta}}) \neq \boldsymbol{\beta}$$

mean of the distribution of all possible sample parameters

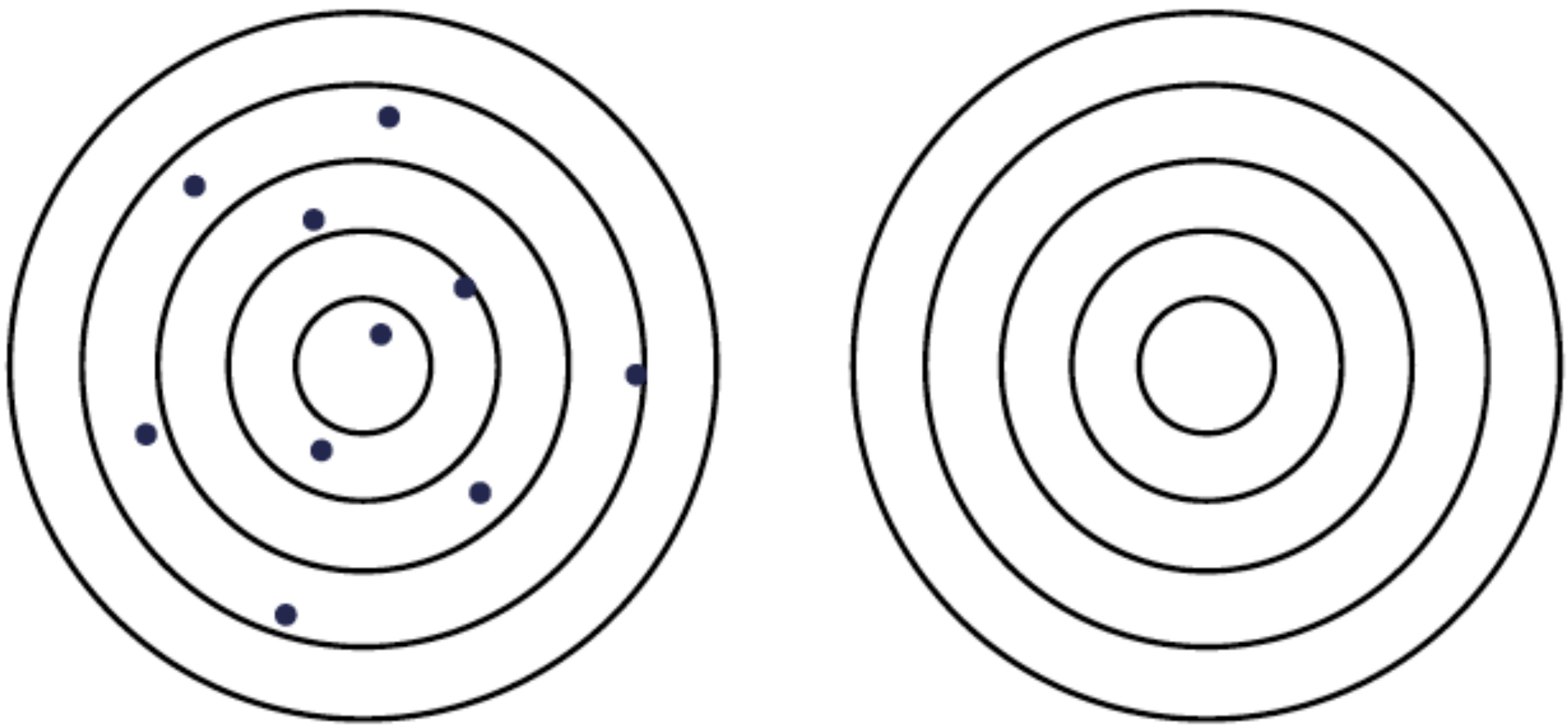
population "truth"

Biased Regression

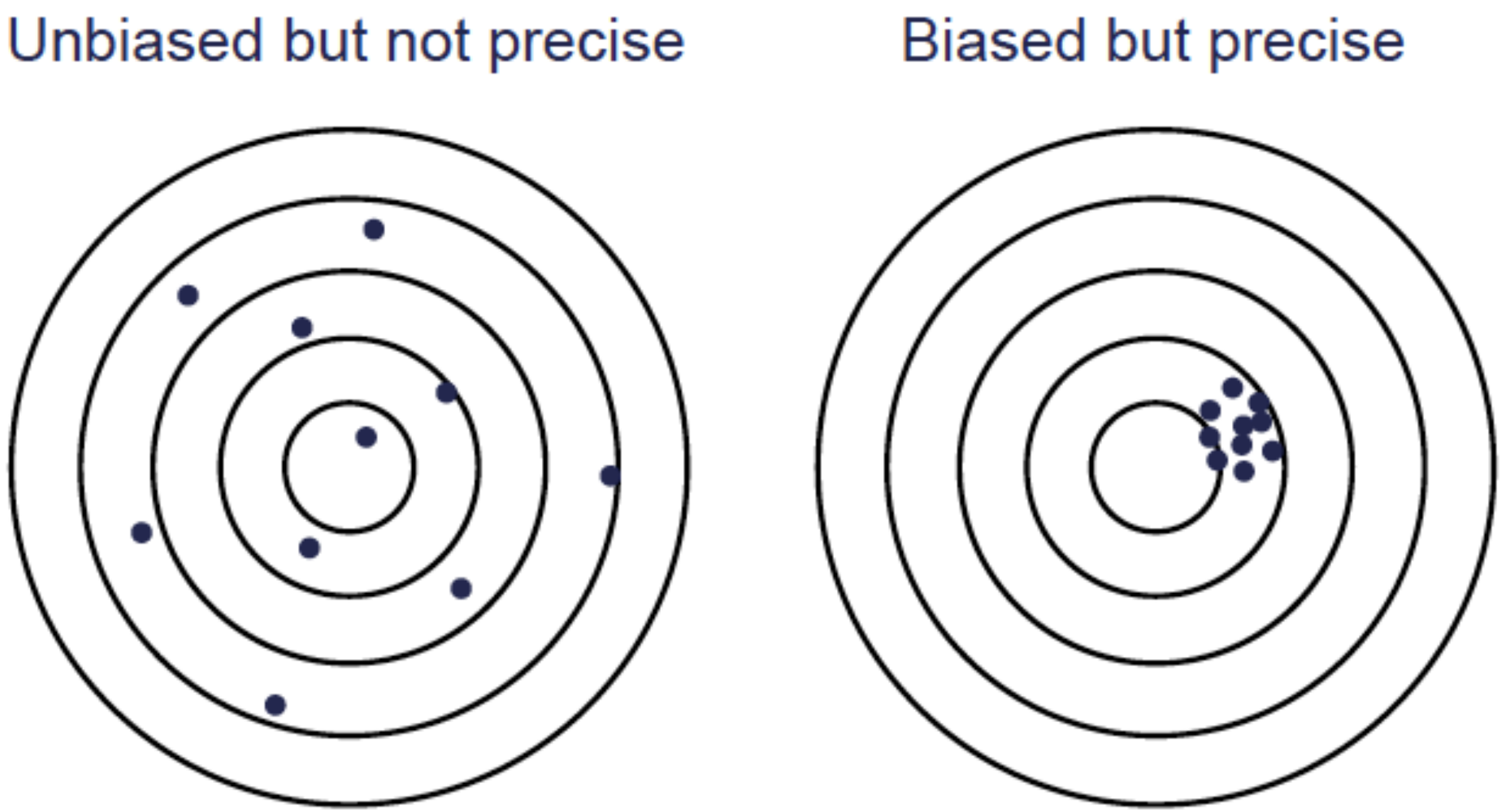


Biased Regression

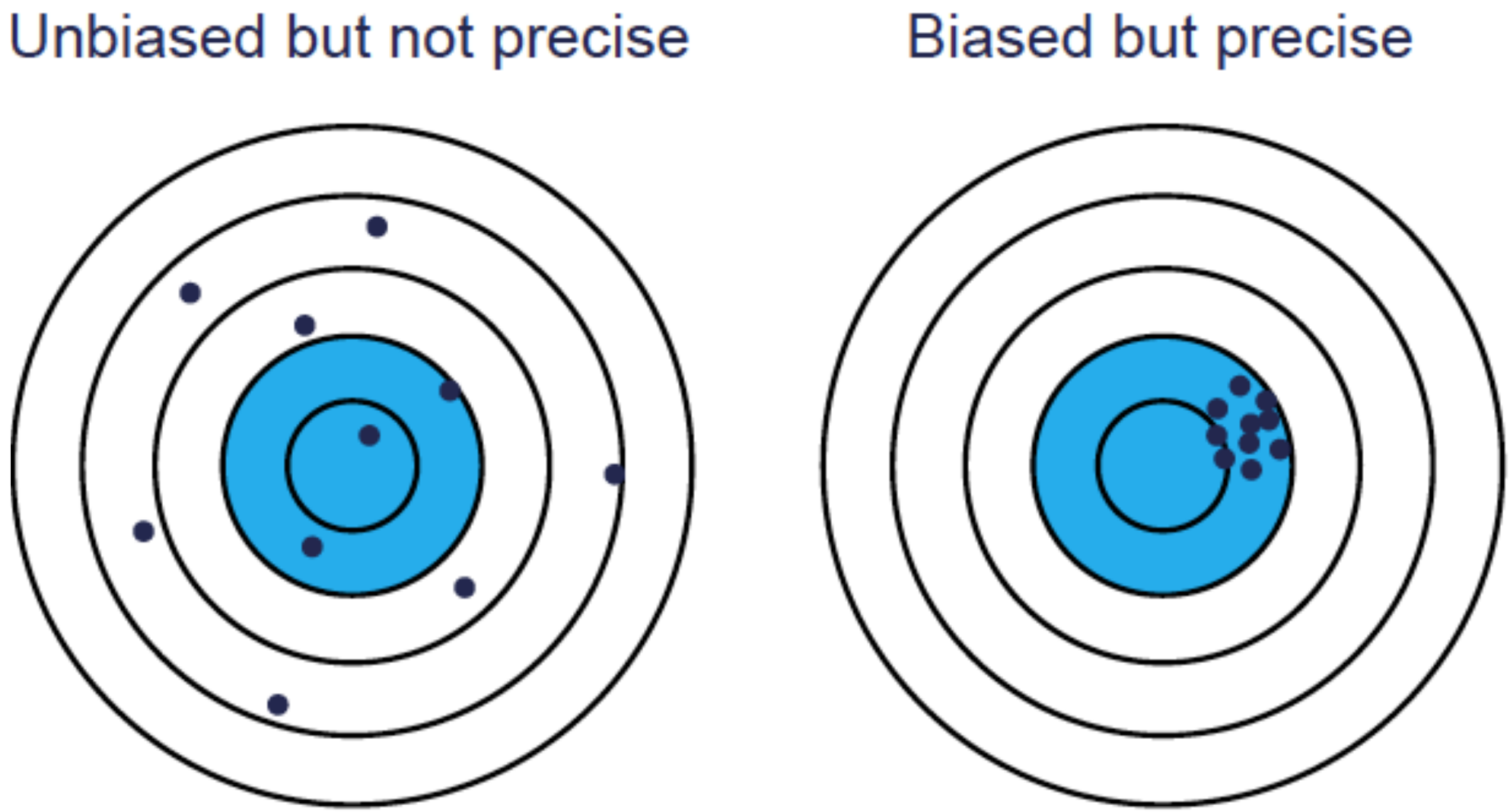
Unbiased but not precise



Biased Regression



Biased Regression



Biased Regression

- What do you lose?
  - Statistical testing of significance
- What do you gain?
  - (Hopefully) Predictive accuracy on validation data

Dealing with Multicollinearity

- PCA gives us a new representation of our data that is completely uncorrelated.

- HOWEVER, **using all the principal components does not solve** the underlying problem of **multicollinearity**. It just hides it through rotation.

- **Must drop some components** to solve severe multicollinearity.

Principal Components Regression

- Want to model target, $\mathbf{y}$ as a function of $\mathbf{x}$'s:

$$\mathbf{y} = \beta_0 + \beta_1 \mathrm{x}_1 + \beta_2 \mathrm{x}_2 + \ldots + \beta_p \mathrm{x}_p + \epsilon$$

- Use the principal components (the scores for each observation) as your new predictor variables.

$$\mathbf{y} = \alpha_0 + \alpha_1 \mathrm{prin}_1 + \alpha_2 \mathrm{prin}_2 + \ldots + \alpha_k \mathrm{prin}_k + \epsilon$$

Principal Components Regression

- Use the fact that the principal components are linear combinations of your original variables to get back to the β's (for interpretation):

$$\text{prin1} = \mathbf{v}_{11}\mathbf{x}_1 + \mathbf{v}_{21}\mathbf{x}_2 + \ldots + \mathbf{v}_{p1}\mathbf{x}_p$$

Entries from the eigenvectors (loadings)

Replacement math for correlation PCA

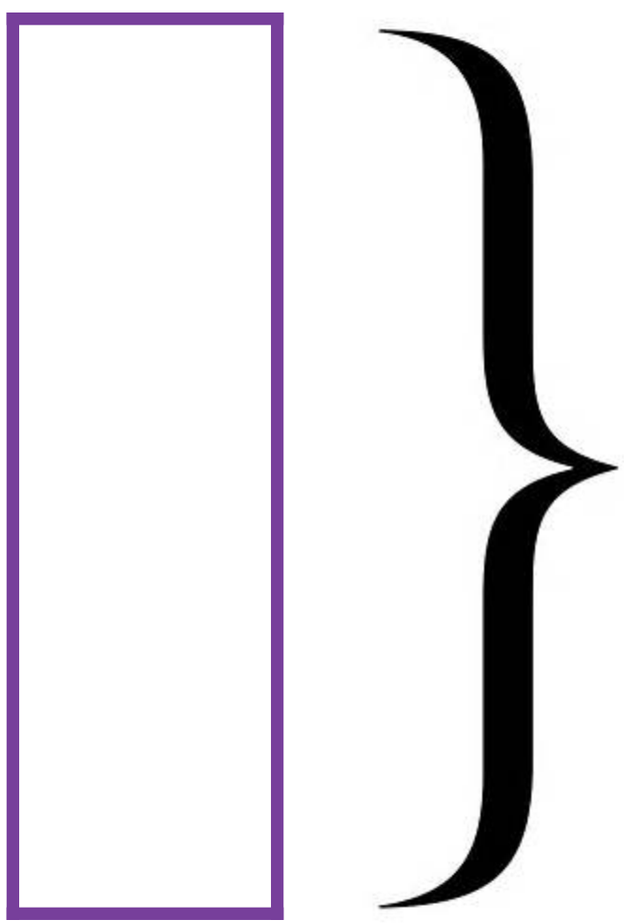
$$\tilde{\mathbf{y}} = \frac{\mathbf{y} - \bar{\mathbf{y}}}{s_y} \quad \tilde{\mathbf{x}} = \frac{\mathbf{x} - \bar{\mathbf{x}}}{s_x}$$

$$\begin{aligned}
\tilde{\mathbf{y}} &= \alpha_1 PC_1 + \alpha_2 PC_2 + \cdots + \alpha_p PC_p + \epsilon \\
PC_j &= \mathbf{V}_{1j}\tilde{\mathbf{x}}_1 + \mathbf{V}_{2j}\tilde{\mathbf{x}}_2 + \cdots + \mathbf{V}_{pj}\tilde{\mathbf{x}}_p \\
\mathbf{y} &= \beta_0 + \beta_1 \mathbf{x}_1 + \cdots + \beta_p \mathbf{x}_p + \epsilon \\
\beta_j &= \frac{s_y}{s_{x_j}}(\mathbf{V}_{j1}\alpha_1 + \mathbf{V}_{j2}\alpha_2 + \cdots + \mathbf{V}_{jp}\alpha_p) \\
\beta_0 &= \bar{\mathbf{y}} - \beta_1 \bar{\mathbf{x}}_1 - \beta_2 \bar{\mathbf{x}}_2 - \cdots - \beta_p \bar{\mathbf{x}}_p
\end{aligned}$$

Choosing the number of components for PCR

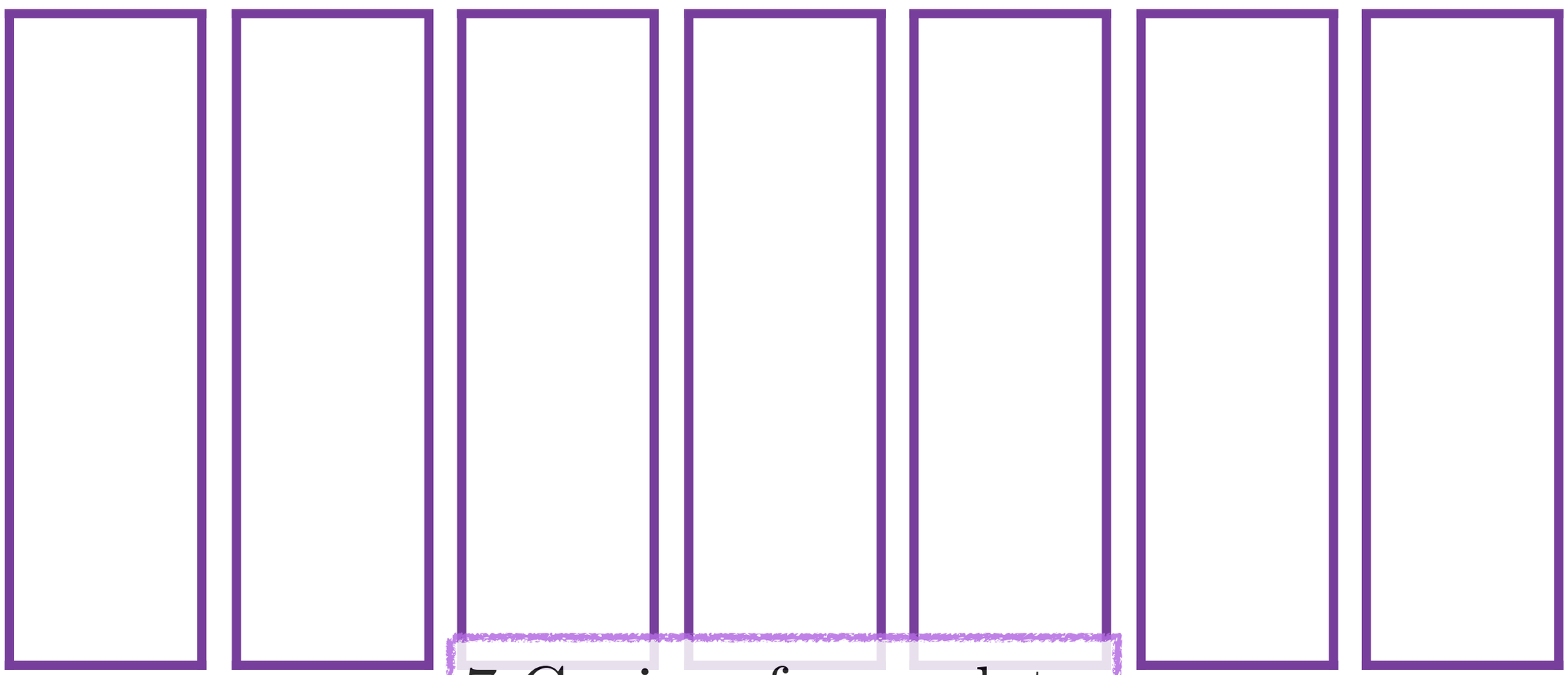
(cross-validation)

Cross Validation



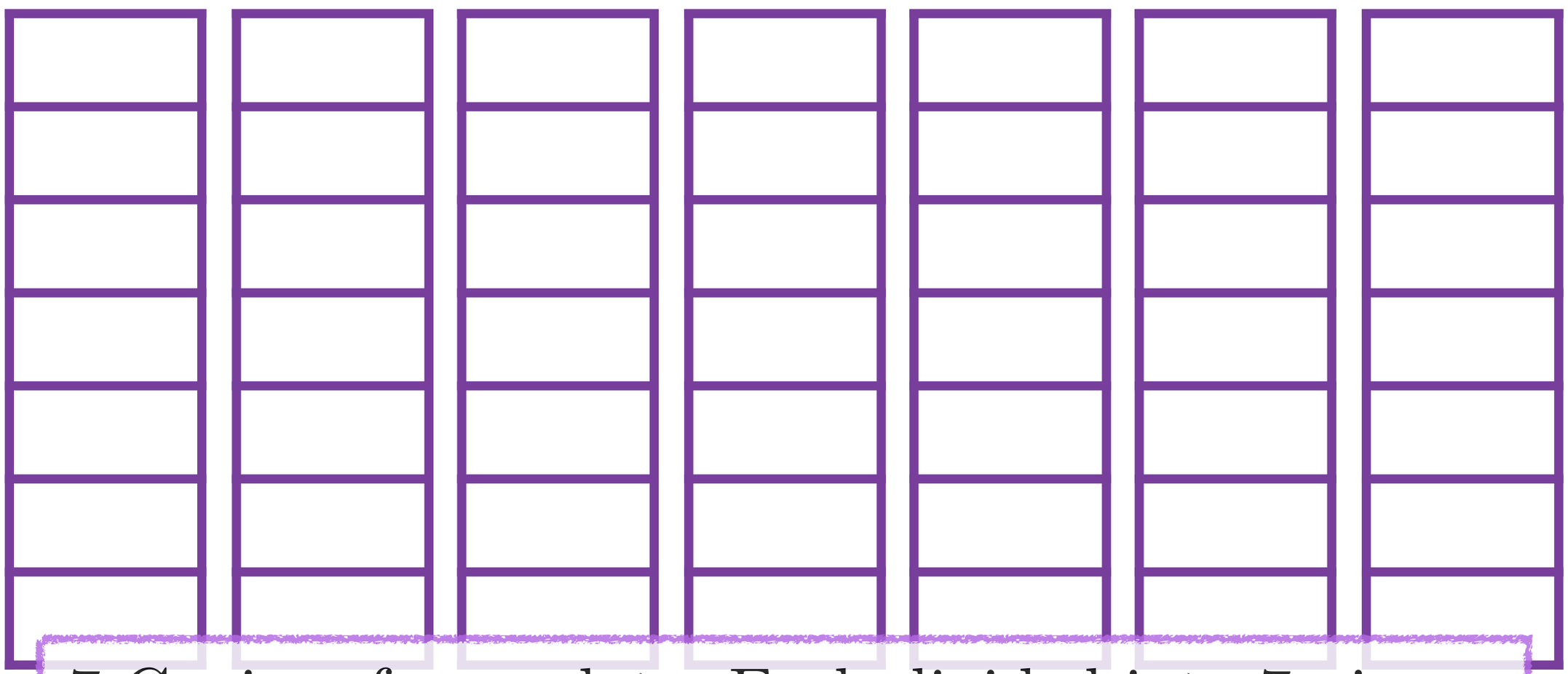
Your data. Many observations.

Cross Validation



7 Copies of your data.

Cross Validation

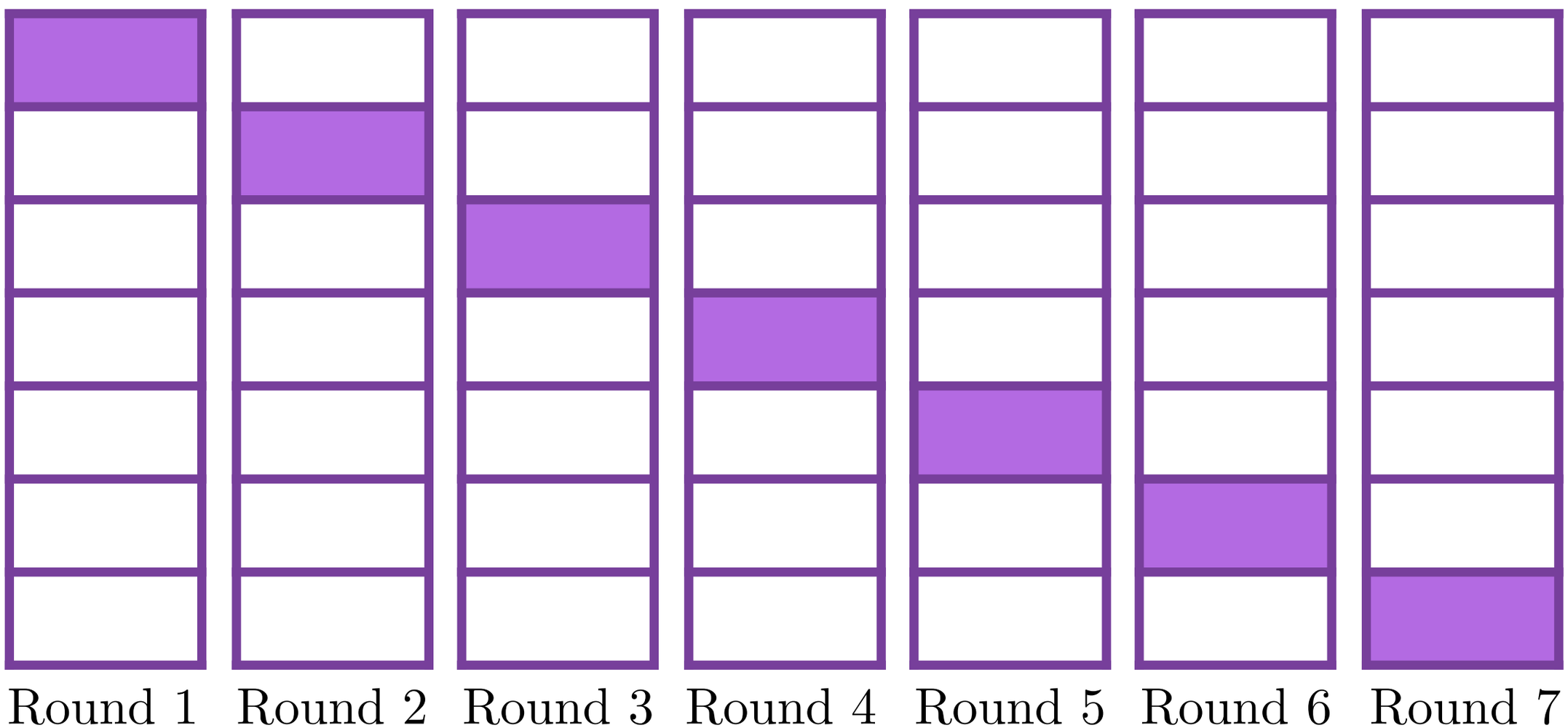


7 Copies of your data. Each divided into 7 pieces.

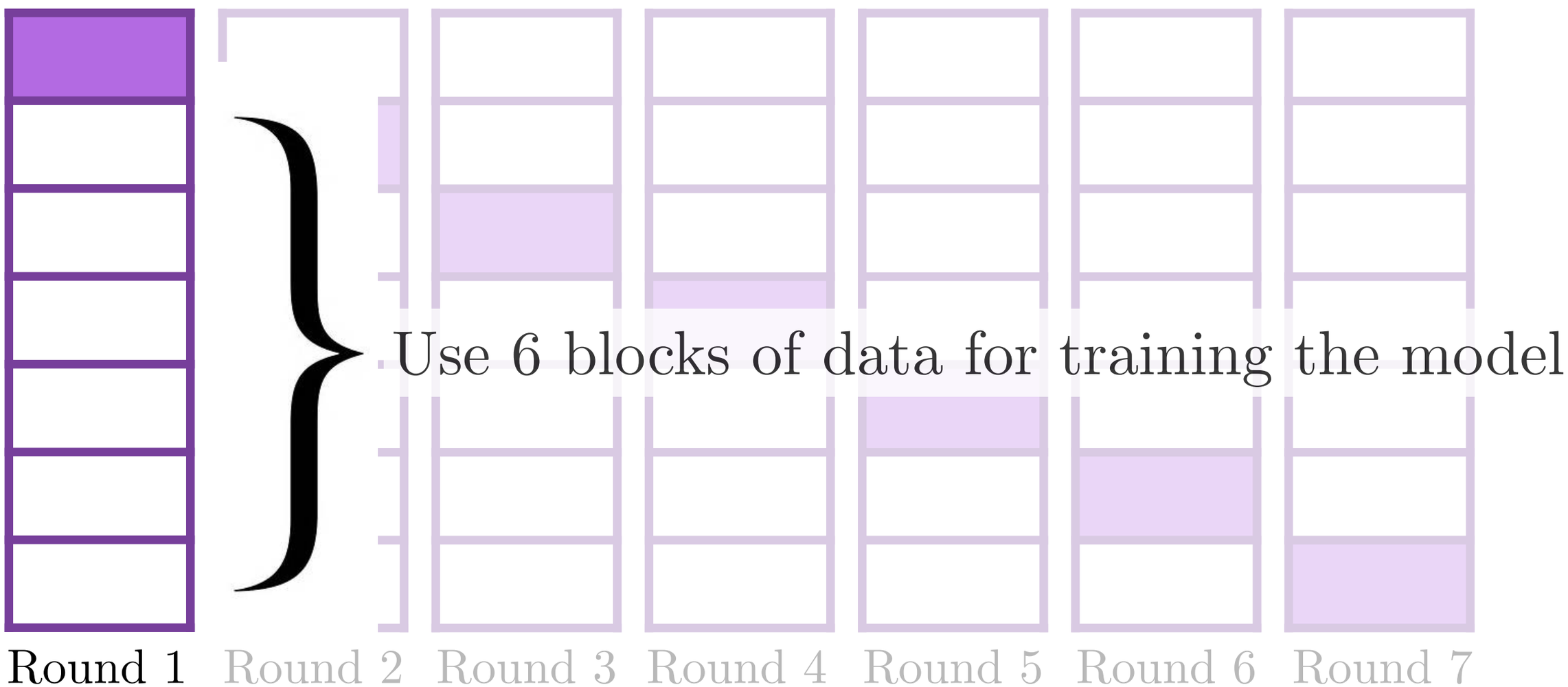
Round 1 Round 2 Round 3 Round 4 Round 5 Round 6 Round 7

Cross Validation

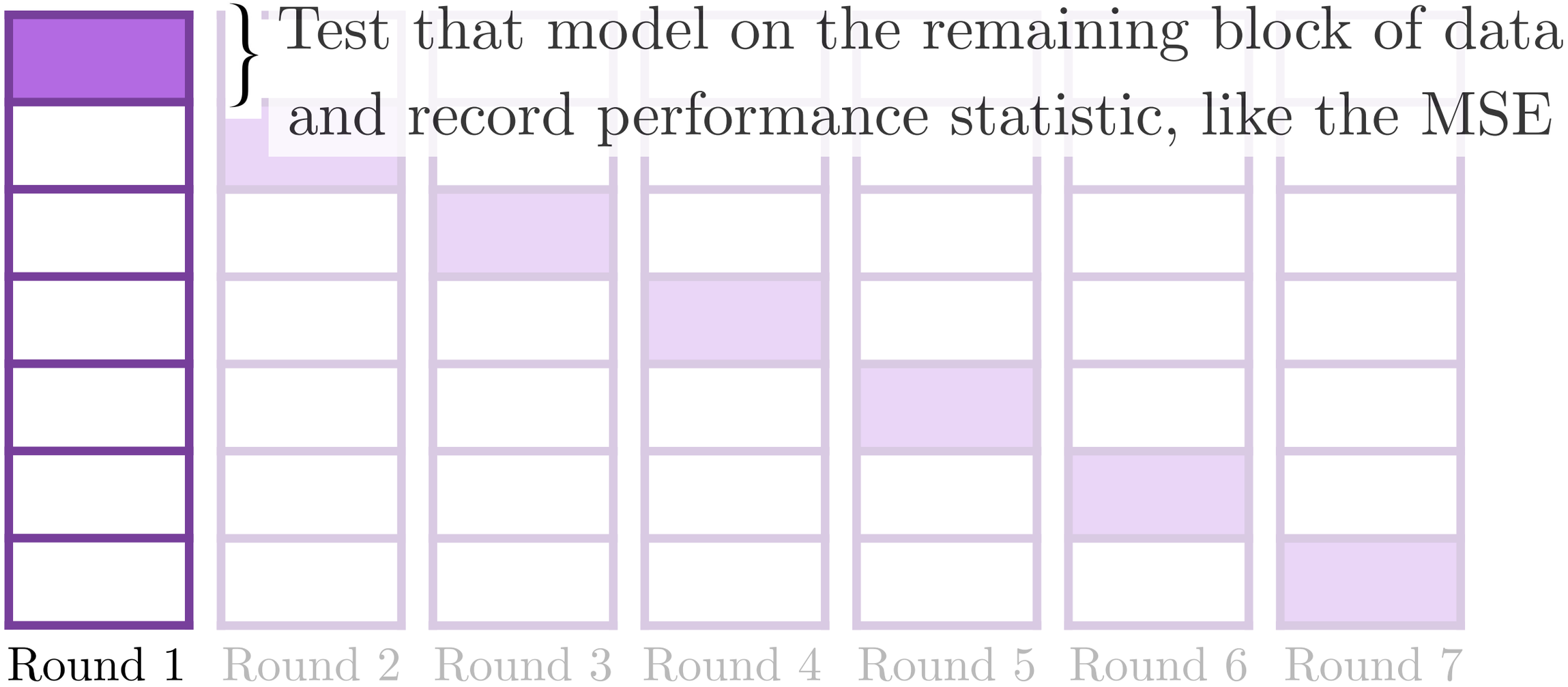
7-Fold Cross Validation.



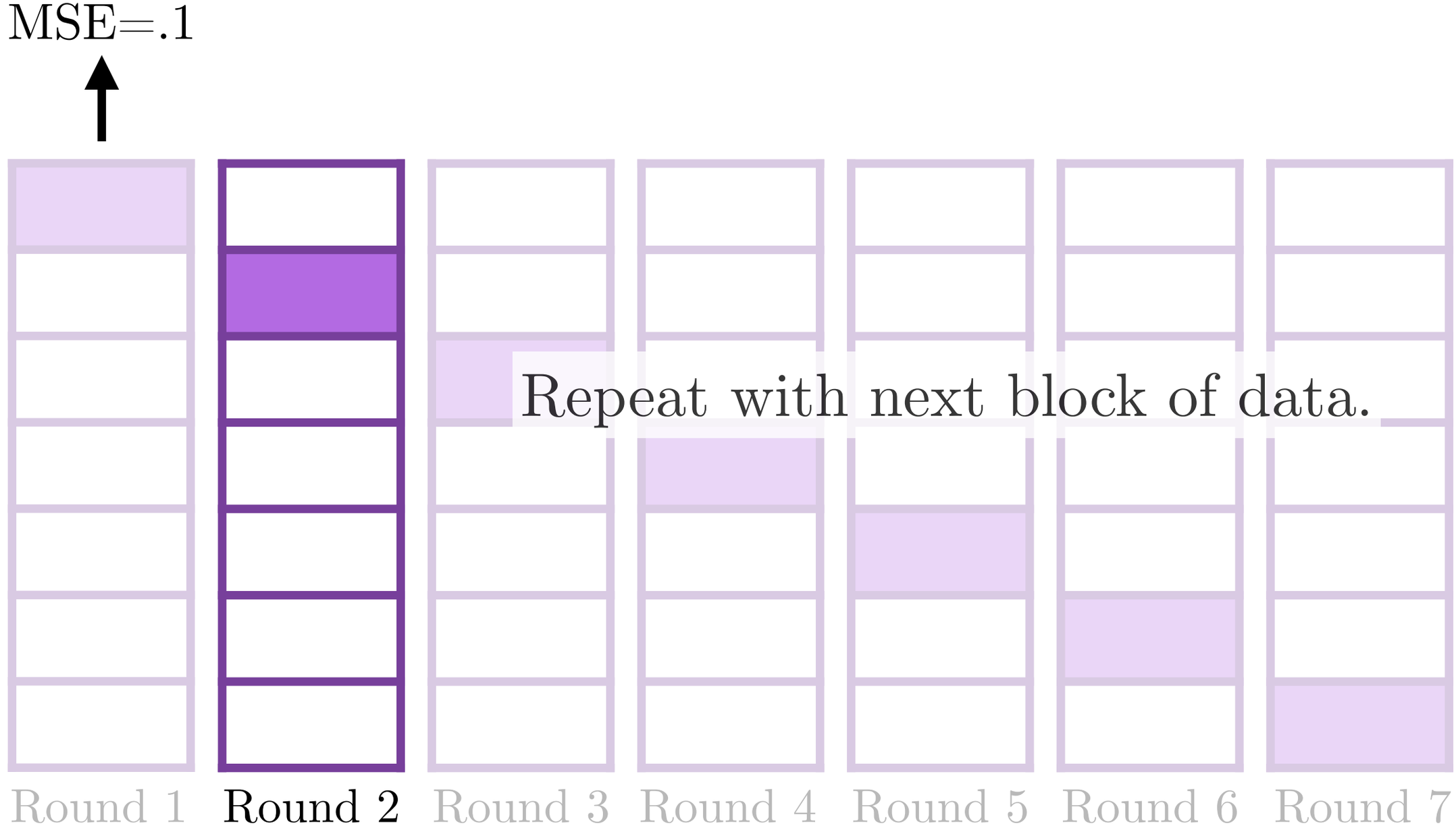
Cross Validation



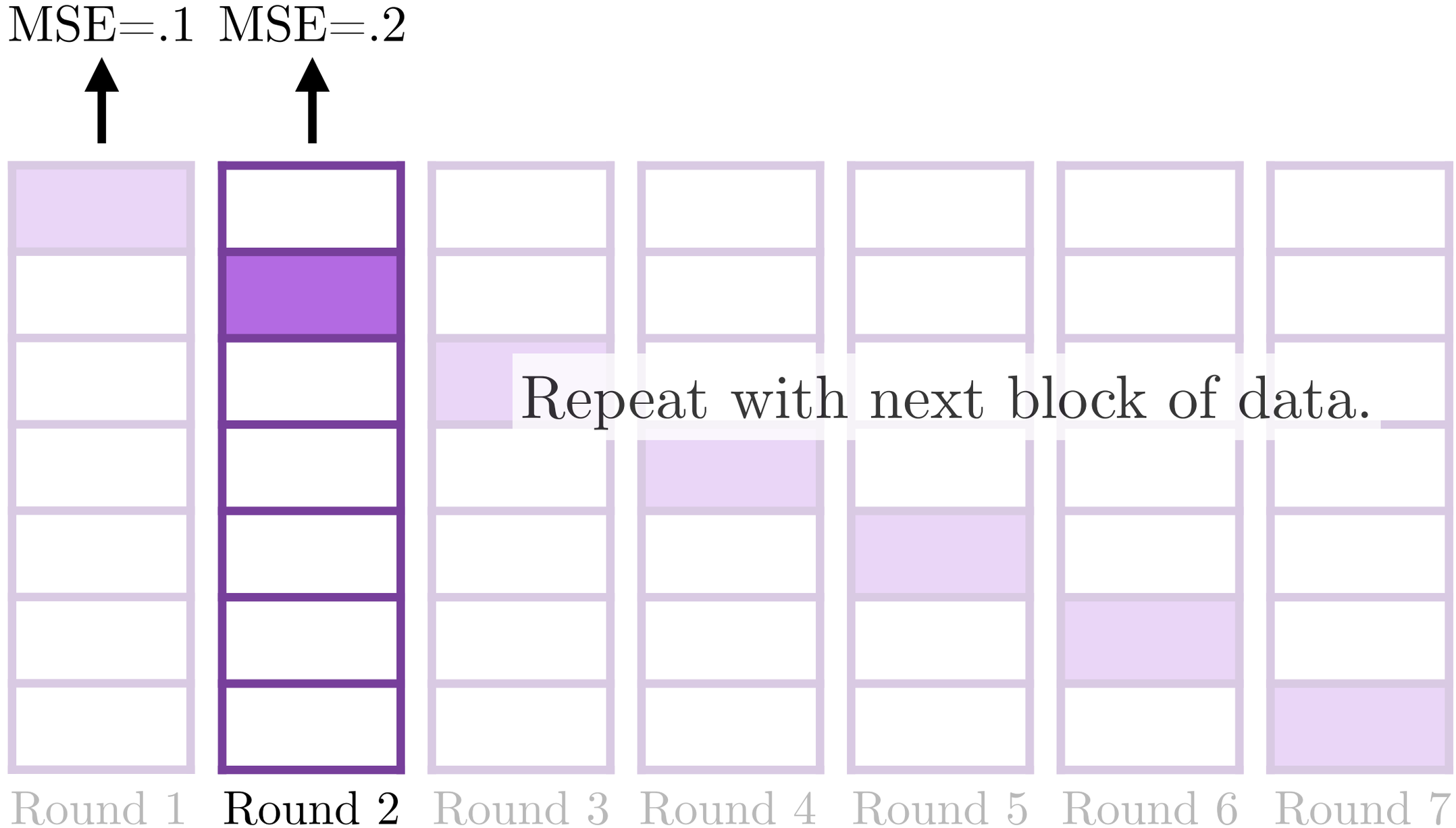
Cross Validation



Cross Validation

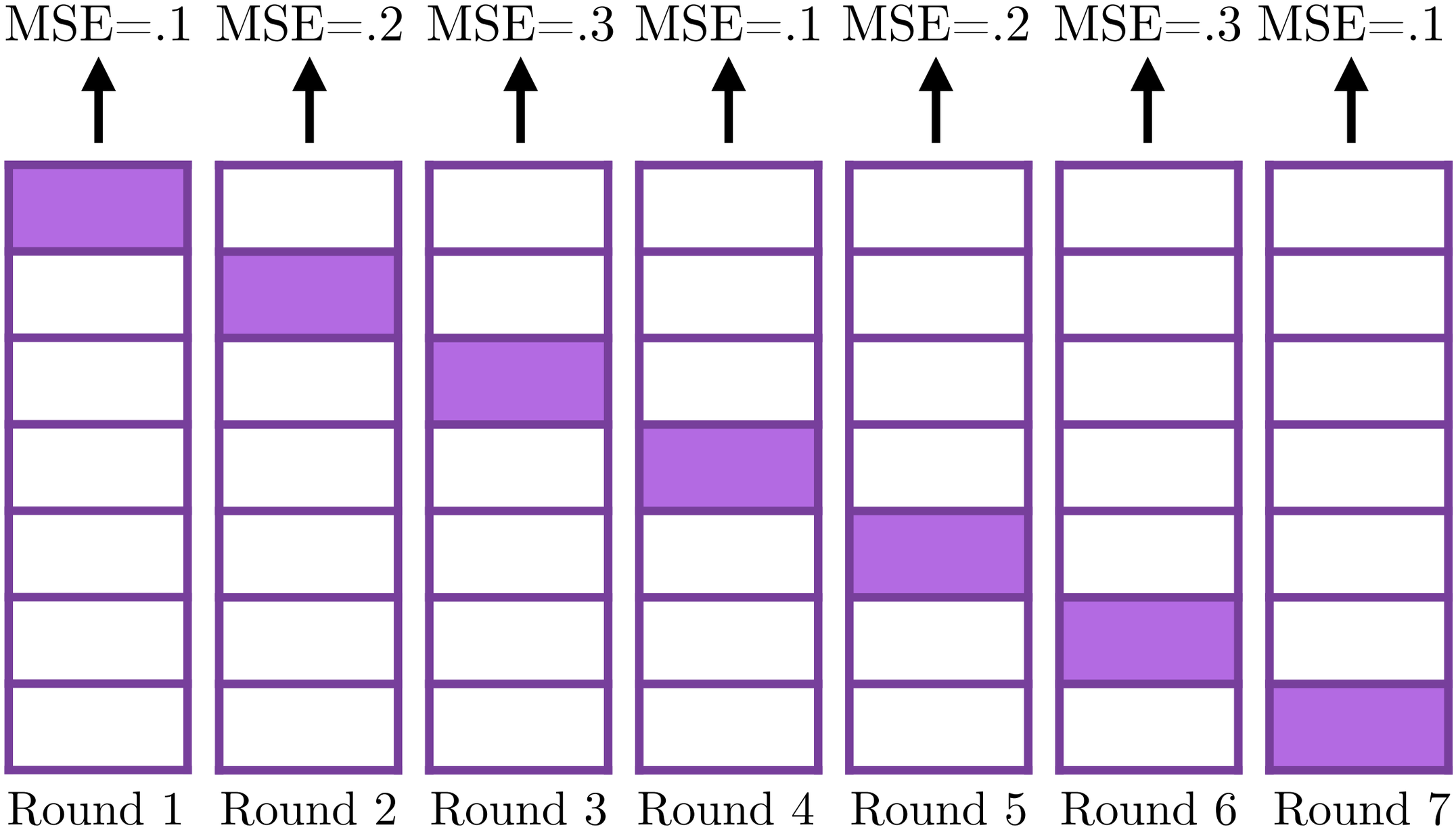


Cross Validation



Cross Validation

(At Completion)



Cross Validation

(At Completion)

MSE=.1 MSE=.2 MSE=.3 MSE=.1 MSE=.2 MSE=.2 MSE=.1

Average out-of-sample MSE for the trained model:

$$(.1+.2+.3+.1+.2+.2+.1)/7 = \mathbf{0.17}$$

We'd repeat this process, training r different models using 1, 2, 3, ... , r principal components.

In this example, each of the r models would be trained and tested 7 different times (once at each CV iteration).

Applying Principal Component Regression

Baseball data: Inputs

AtBat: Number of times at bat in 1986
Hits: Number of hits in 1986
HmRun: Number of home runs in 1986
Runs: Number of runs in 1986
RBI: Number of runs batted in in 1986
Walks: Number of walks in 1986
Years: Number of years in the major leagues
CAtBat: Number of times at bat during his career
CHits: Number of hits during his career
CHmRun: Number of home runs during his career
CRuns: Number of runs during his career
CRBI: Number of runs batted in during his career
CWalks: Number of walks during his career
League: A factor with levels A and N indicating player's league at the end of 1986
Division: A factor with levels E and W indicating player's division at the end of 1986
PutOuts: Number of put outs in 1986
Assists: Number of assists in 1986
Errors: Number of errors in 1986
NewLeague: A factor with levels A and N indicating player's league at the beginning of 1987

Goal: Predict 1987 *Salary*

Principal Component Regression

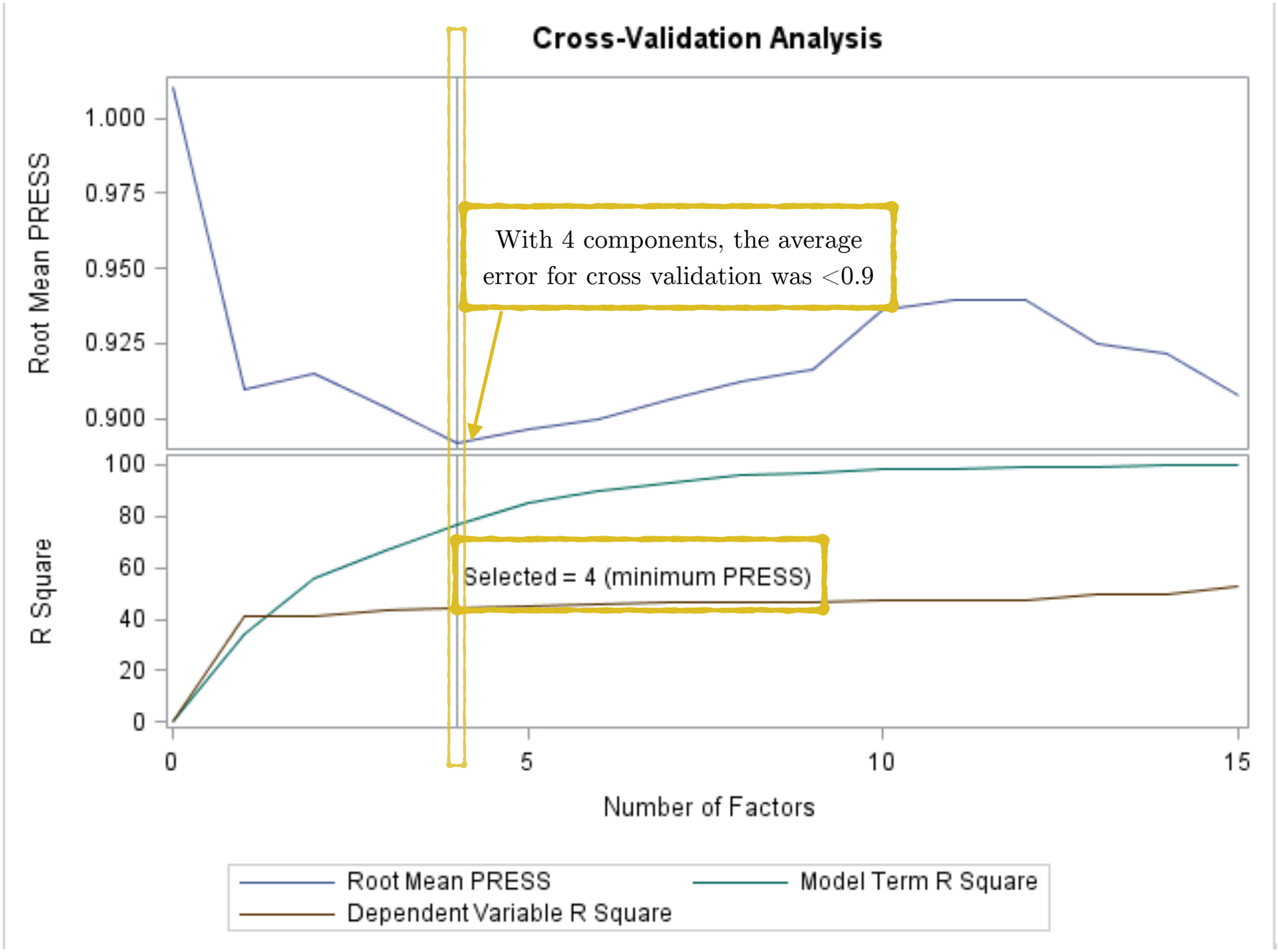
principal components regression

CV folds

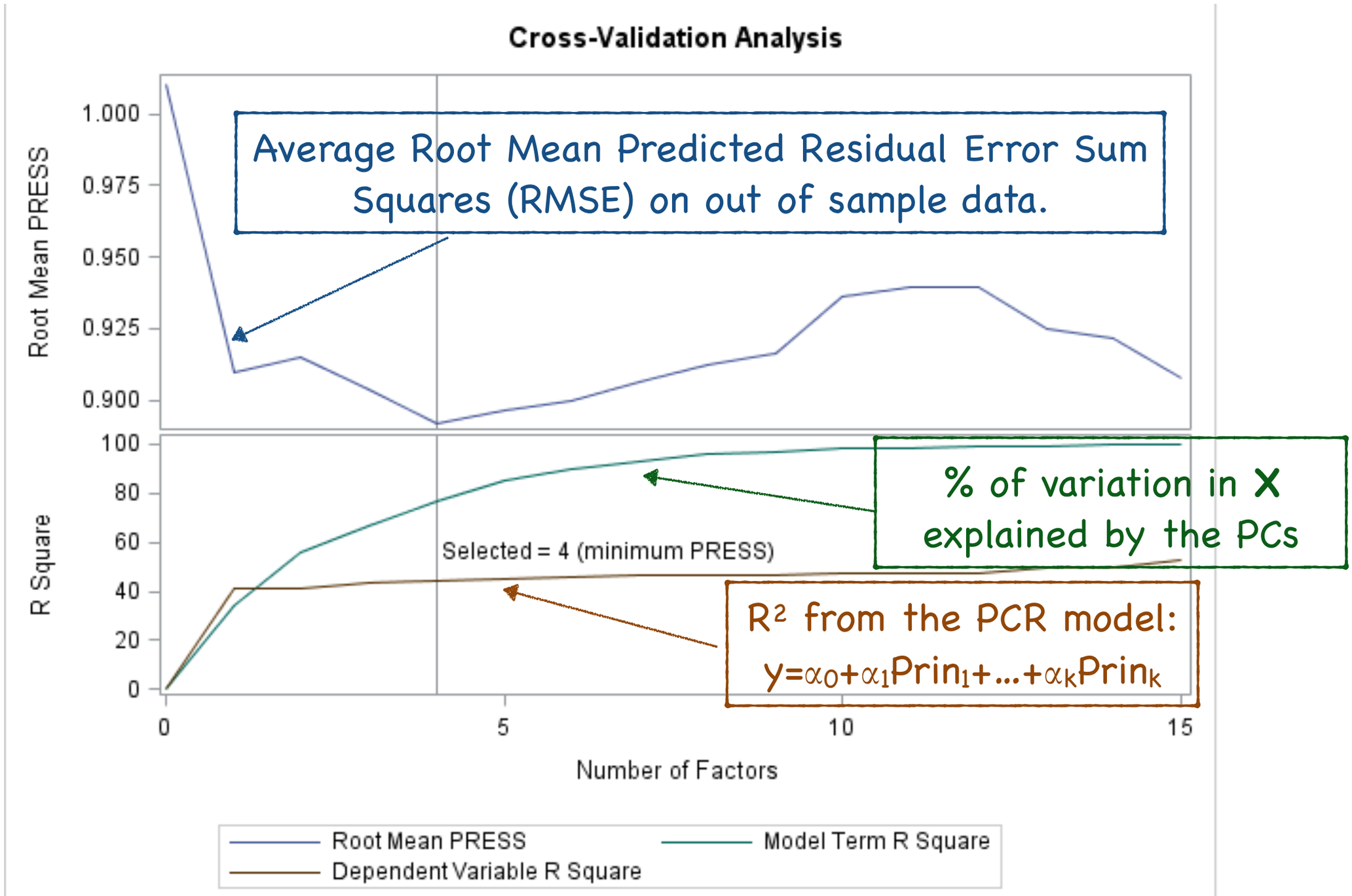
random seed

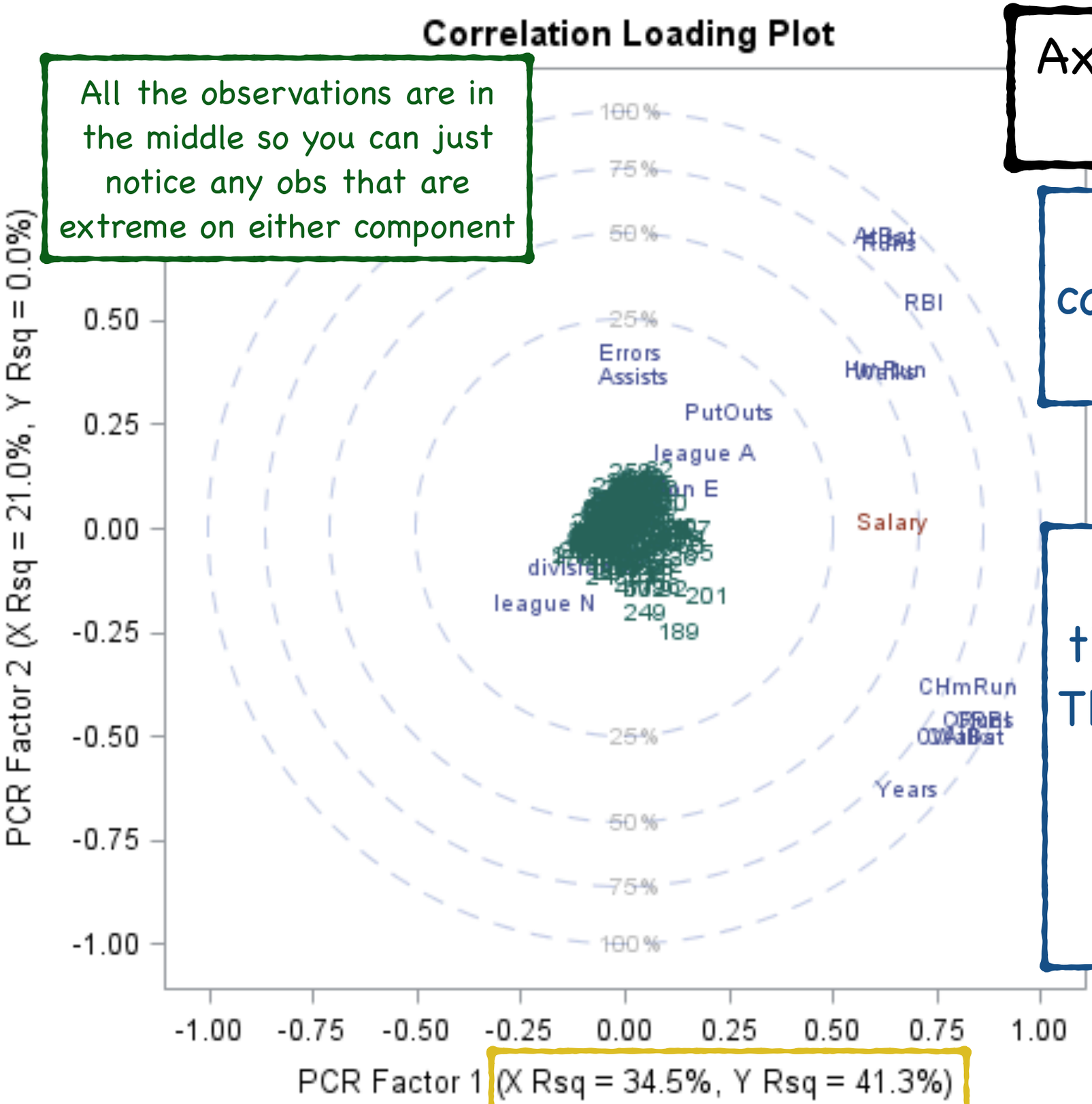
```
proc pls data=hitters method=pcr
          cv=random(niter=10 seed=13);
class league division;
model salary = years walks runs rbi putouts league hmrun
               hits errors division cwalks cruns crbi chmrun
               catbat atbat assists ;
run;
```

CV Result: Use 4 components to minimize out-of-sample error.



CV Chart Interpretation





All the observations are in the middle so you can just notice any obs that are extreme on either component

Axes represent 1st and 2nd principal components

(x,y) coordinates give correlation of each variable with each of the PCs

"radial" coordinate gives the variables' **communality.** This is the R^2 for predicting the variable using the 2 principal components.

(see comments in Factor Analysis in-class SAS code using IPIP data)

X Rsq refers to % of variance explained by this PC.

Y Rsq refers to the R^2 for predicting y with just this PC.

Finalize Model

specify number of components from CV

```
proc pls data=hitters method=pcr nfac=4;
class league division;
model salary = years walks runs rbi putouts league hmrun
               hits errors division cwalks cruns crbi chmrun
               catbat atbat assists / solution;
run;
```

Give us parameters in terms of original variables

Finalize Model

```
proc pls data=hitters method=pcr nfac=4;
class league division;
model salary = years walks runs rbi putouts league hmrun
               hits errors division cwalks cruns crbi chmrun
               catbat atbat assists / solution;
run;
```

Parameter Estimates	
	Salary
Intercept	-92.35271090
Years	6.31662935
Walks	1.39791674
Runs	1.13111785
RBI	1.19183885
PutOuts	0.04771647
league A	-22.82063342
league N	22.82063342
HmRun	2.49288572
Hits	0.65652664
Errors	1.41196754
division E	68.60920143
division W	-68.60920143
CWalks	0.13896648
CRuns	0.11692262
CRBI	0.11655947
CHmRun	0.41285542
CAtBat	0.01636646
AtBat	0.20028408
Assists	0.07016641

What? No testing? 😩

- Would it really make sense?
- How much of salary is due to home runs this year vs. hits this year? career home runs vs home runs this year?

A Big Data Example where PCR destroys OLS

The Big Data Set

- 500,000 observations
- 120 numeric input variables labelled **v1-v120**
- 1 numeric target variable labelled **target**
- 1 variable indicating test set labelled **test**
  (i.e. test=1 for test obs, test=0 for training obs)
- Source of the data? top-secret.

1.) Implement OLS Regression

2.) Score Test Data

```
proc reg data=public.bigdatapcr(datalimit=all where=(test=0)) outest=OLS;
    OLS: model target=v1--v120 /vif;
    run;
```

Variance **I**nflation **F**actors

Data Exceeded Size Limit. Override.

Only use training observations

Output Parameter Estimates for Proc Score

```
proc score data=public.bigdatapcr(datalimit=all where=(test=1)) score=OLS
        type=parms predict out=out;
    var v1--v120;
run;
```

1.) Implement OLS Regression

2.) Score Test Data

```
proc reg data=public.bigdatapcr(datalimit=all where=(test=0)) outest=OLS;
    OLS: model target=v1--v120 /vif;
    run;
```

Output scored test data to dataset **out**

Only use test observations

Input Parameter Estimates from Proc Reg

```
proc score data=public.bigdatapcr(datalimit=all where=(test=1)) score=OLS
        type=parms predict out=out;
    var v1--v120;
run;
```

VIFs 😳

Variable	DF	Parameter Estimate	Standard Error	t Value	Pr > |t|	Variance Inflation
Intercept	B	2.17843	0.08203	26.56	<.0001	0
v1	B	-0.00711	0.02301	-0.31	0.7571	5.84603
v2	B	-0.00597	0.00855	-0.70	0.4850	1.16118
v3	B	-0.00783	0.00305	-2.57	0.0103	1.20244
v4	B	-0.00459	0.00089012	-5.15	<.0001	1.10480
v5	B	-0.00001596	0.00001481	-1.08	0.2810	1.94898
v6	B	-0.00002410	0.00002048	-1.18	0.2391	3.28746
v7	B	0.00002350	0.00002113	1.11	0.2660	3.33584
v8	B	0.00002434	0.00001323	1.84	0.0659	1.45397
v9	B	0.25536	0.01234	20.69	<.0001	3.01983
v10	B	0.63231	0.05127	12.33	<.0001	19.55051
v11	B	0.00004157	0.00003499	1.19	0.2349	1.90771
v12	B	0.00003607	0.00003431	1.05	0.2932	1.82003
v13	B	-0.00007442	0.00002376	-3.13	0.0017	1.47444
v14	B	-0.00054214	0.00007702	-7.04	<.0001	1.96191
v15	B	0.06254	0.01455	4.30	<.0001	1.52030
v16	B	0.04172	0.05750	0.73	0.4681	22.08305
v17	B	-0.00000308	0.00001452	-0.21	0.8320	1.61480
v18	B	-0.01196	0.00683	-1.75	0.0798	28800
v19	B	0.04948	0.00553	8.95	<.0001	18897
v20	B	-0.03926	0.00807	-4.86	<.0001	40244
v21	B	1.75109E-15	5.93136E-15	0.30	0.7678	1.00342
v22	B	-0.00472	0.00190	-2.48	0.0130	7.19173
v23	B	-0.00226	0.00150	-1.51	0.1320	8.36047
v24	B	-0.00926	0.00161	-5.77	<.0001	26.64846
v25	B	-0.00115	0.00072775	-1.57	0.1156	35.95481
v26	B	-0.00021726	0.00065726	-0.33	0.7410	58.42737
v27	B	0.02181	0.00239	9.13	<.0001	1128.53911
v66	B	-0.06152	0.01929	-3.19	0.0014	126.54519
v67	B	0.00011494	0.00001435	8.01	<.0001	1.73115
v68	B	-0.01006	0.00127	-7.95	<.0001	1.87201
v69	B	0.00615	0.00257	2.40	0.0165	69.53026
v70	B	0.00969	0.00113	8.61	<.0001	23.53995
v71	B	-0.02213	0.00196	-11.28	<.0001	198.17739
v72	B	-0.01213	0.00167	-7.28	<.0001	237.71557
v73	B	0.00961	0.00116	8.27	<.0001	153.42279
v74	B	-0.02312	0.00190	-12.19	<.0001	735.58916
v75	B	0.00060135	0.00054701	1.10	0.2716	160.79102
v76	B	0.00197	0.00017361	11.32	<.0001	25.00046
v77	B	0.31646	0.03731	8.48	<.0001	2.81921
v78	B	2.07168E-12	1.13209E-12	1.83	0.0673	5.95465
v79	B	-0.00485	0.00102	-4.75	<.0001	2.09786
v80	B	-0.01972	0.01003	-1.97	0.0494	1.00647
v81	B	-0.02376	0.00115	-20.72	<.0001	71.23798
v82	B	0.41100	0.04509	9.12	<.0001	3.56955
v83	B	0.02198	0.00122	18.01	<.0001	49.17466

3) Compute PCR:

a) Find optimal number of components

b) Use optimal number of components

```
proc pls data=public.bigdatapcr(datalimit=all where=(test=0))
         method=pcr
         cv=random(niter=2 seed=100816);
    model target=v1--v128;
run;
```

Only use training observations

```
ods output parameterestimates=PCR_estimates;
proc pls data=public.bigdatapcr(datalimit=all where=(test=0)) method=pcr
        nfac=15;
    model target=v1--v120/solution;
run;
```

3) Compute PCR:

a) Find optimal number of components

b) Use optimal number of components

```
proc pls data=public.bigdatapcr(datalimit=all where=(test=0))
         method=pcr
         cv=random(niter=2 seed=100816);
    model target=v1--v128;
run;
```

Only use training observations

Use ods output and solution option to get parameter estimates for use in proc score

```
ods output parameterestimates=PCR_estimates;
proc pls data=public.bigdatapcr(datalimit=all where=(test=0)) method=pcr
         nfac=15;
    model target=v1--v120/solution;
run;
```

4) Score test data with PCR model:

a) Create score table from ods output

```
proc transpose data=PCR_estimates out=PCR_estimates;
    id rowname;
run;

data PCR_estimates;
    set PCR_estimates;
    _type_='parms';
    _model_='PCR';
    _depvar_='Overall';
    drop _name_;
run;
```

Compare the table **work.OLS** with **work.PCR_estimates**

These two procedures transform **work.PCR_estimates** to match the structure of **work.OLS**

4) Score test data with PCR model:

b) score test data with proc score

```
proc score data=out score=PCR_estimates type=parms predict out=out2;
    var v1--v120;
run;
```

Output from first proc score contained original data plus OLS predicted values. By using that version of data we'll have both predictions on same table

That final table called **out2**

5) Compare the two models

a) via R^2

b) via RMSE

And the winner is: PCR

```
proc corr data=out2;
    var PCR ols;
    with target;
run;
```

Pearson Correlation Coefficients, N = 200000 Prob > |r| under H0: Rho=0		
	PCR	OLS
target	0.38610 <.0001	0.35339 <.0001

```
proc sql;
    select sqrt(mean((OLS - target)**2)) as OLS,
           sqrt(mean((PCR - target)**2)) as PCR
    from out2;
run;
```

OLS	PCR
0.391911	0.369571

Partial Least Squares (PLS)

Supervised vs. Unsupervised

- PCA is an **unsupervised** method of analysis

  The directions of maximal variance do not take into account a target/response variable $\mathbf{y}$

- PLS is a **supervised** alternative to PCA

  The directions are drawn to not only best summarize the $\mathbf{X}$ data but also to best predict a target/response variable $\mathbf{y}$.

Partial Least Squares

- First PLS direction $\mathbf{z}_1$ is a linear combination of predictor variables where the coefficient of $\mathbf{x}_j$ is the simple linear regression coefficient of $\mathbf{y}$ on $\mathbf{x}_j$
  - $=>$ highest weight on variables most correlated with y.
- Then, data is orthogonalized and next direction drawn in same manner. Repeat until $\mathbf{p}$ components.

Partial Least Squares

- Popular in some scientific disciplines, particularly when more than one target/response variable.
- In practice, does not perform better than PCR or Ridge regression (...coming in Fall 3)

Partial Least Squares on Big Data Example

1) Implement PLS

a) Compute optimal number of components

b) Create model

Just change method to PLS!

```
proc pls data=public.bigdatapcr(datalimit=all where=(test=0))
         method=pls
          cv=random(niter=2 seed=100816);
     model target=v1--v128;
 run;
```

2-fold Cross-Validation to determine # Components

```
ods output parameterestimates=PLS_estimates;
proc pls data=public.bigdatapcr(datalimit=all where=(test=0))
         method=pls
         nfac=15;
    model target=v1--v120/solution;
run;
```

Same ODS output trick to get scoring coefficients

2) Score test data with PLS model

a) Create score table from ods output

```
proc transpose data=PLS_estimates out=PLS_estimates;
    id rowname;
run;

data PLS_estimates;
    set PLS_estimates;
    _type_='parms';
    _model_='PLS';
    _depvar_='Overall';
    drop _name_;
run;
```

Compare the table **work.OLS** with **work.PLS_estimates**

These two procedures transform **work.PLS_estimates** to match the structure of **work.OLS**

4) Score test data with PLS model:

b) score test data with proc score

```
proc score data=out2 score=PLS_estimates type=parms predict out=out3;
    var v1--v120;
run;
```

Output from second proc score contained original data plus OLS predicted values and PCR predicted values. By using that version of data we'll have all 3 predictions on same table

That final table called **out3**

5) Compare the three models

a) via R^2

b) via RMSE

And the winner is: PCR. still.

```
proc corr data=out2;
    var PCR ols;
    with target;
run;
```

Pearson Correlation Coefficients, N = 200000 Prob > |r| under H0: Rho=0			
	PCR	OLS	PLS
target	0.38610 <.0001	0.35339 <.0001	0.36227 <.0001

```
proc sql;
    select sqrt(mean((OLS - target)**2)) as OLS,
           sqrt(mean((PCR - target)**2)) as PCR
    from out2;
run;
```

OLS	PCR	PLS
0.391911	0.369571	0.404463