Week 4 - Partial Least Squares Regression

Please indicate at the top of your assignment whether or not you used any AI tools, such as MS Copilot. If you did use one of these tools, please provide a very brief explanation alongside each answer for how you confirmed the correctness of your solution.

- Used MS Copilot for a couple of questions.

We will reimplement and then explore some of the properties of Cosgrove et al (http://pubs.rsc.org/en/Content/ArticleLanding/2010/MB/b926287c).

In [3]:
```
import scipy as sp, numpy as np
from sklearn.preprocessing import scale, StandardScaler
from sklearn.cross_decomposition import PLSRegression
from sklearn.model_selection import LeaveOneGroupOut, LeaveOneOut
import matplotlib.pyplot as plt
import pandas as pd

data = sp.io.loadmat('wk4_Cosgrove-data.mat', squeeze_me=True)['s']

X = data['X'].item() # the untransformed data matrix (66x102)
Y = data['Y'].item() # the untransformed LDH release at 48hours. (66x1)
phosphoproteins = data['phosphoproteins'].item() # names of phosphoproteins
conditions = data['conditions'].item() # cell array of the 66 conditions
drugList = data['drugList'].item() # description of the drugs used in each of the 66 conditions
drugListToxic = data['drugListToxic'].item() # binary value corresponding to whether drugList[i] is toxic
drugs = data['drugs'].item() # binary matrix mapping which measurements correspond to a drug treatment in drugList
cytokineList = data['cytokineList'].item() # cell array of cytokine treatments
ind4pProtein = data['ind4pProtein'].item() # the column indices corresponding to measurements of the 4 phosphoprote
```

In [4]:
```
print('X:', X.shape, 'Y:', Y.shape)
print('conditions:', conditions)
print('drugList:', drugList)
print('drugListToxic:', drugListToxic)
print('drugs:', drugs.shape)
print('cytokineList:', cytokineList)
print('ind4pProtein:', ind4pProtein)
```

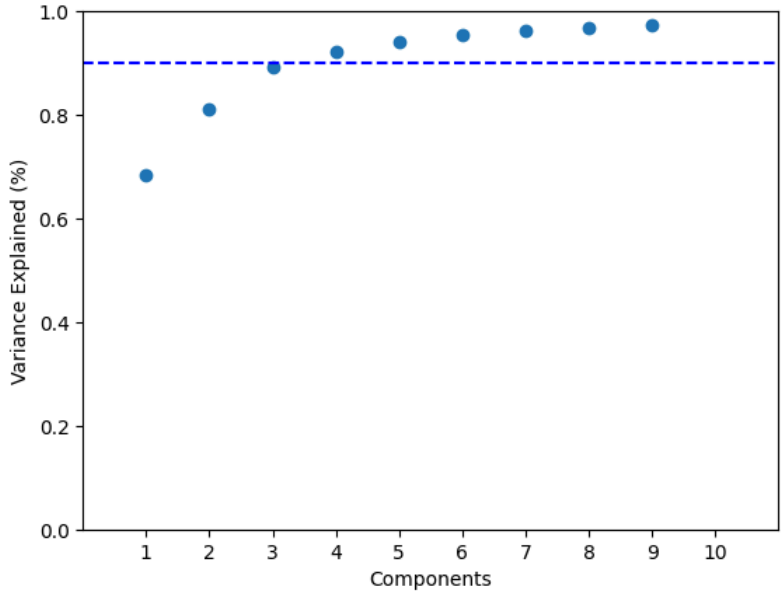
```
X: (66, 102) Y: (66, 5)
conditions: ['DMSO_NoCyt' 'DMSO_IL-1' 'DMSO_LPS' 'DMSO_TNF' 'DMSO_IL-6' 'DMSO_Mix'
 'CIM_NoCyt' 'CIM_IL-1' 'CIM_LPS' 'CIM_TNF' 'CIM_IL-6' 'CIM_Mix'
 'RAN_NoCyt' 'RAN_IL-1' 'RAN_LPS' 'RAN_TNF' 'RAN_IL-6' 'RAN_Mix'
 'LEV_NoCyt' 'LEV_IL-1' 'LEV_LPS' 'LEV_TNF' 'LEV_IL-6' 'LEV_Mix'
 'TRO_NoCyt' 'TRO_IL-1' 'TRO_LPS' 'TRO_TNF' 'TRO_IL-6' 'TRO_Mix'
 'BUS_NoCyt' 'BUS_IL-1' 'BUS_LPS' 'BUS_TNF' 'BUS_IL-6' 'BUS_Mix'
 'NEF_NoCyt' 'NEF_IL-1' 'NEF_LPS' 'NEF_TNF' 'NEF_IL-6' 'NEF_Mix'
 'ASP_NoCyt' 'ASP_IL-1' 'ASP_LPS' 'ASP_TNF' 'ASP_IL-6' 'ASP_Mix'
 'NIM_NoCyt' 'NIM_IL-1' 'NIM_LPS' 'NIM_TNF' 'NIM_IL-6' 'NIM_Mix'
 'CLA_NoCyt' 'CLA_IL-1' 'CLA_LPS' 'CLA_TNF' 'CLA_IL-6' 'CLA_Mix'
 'TEL_NoCyt' 'TEL_IL-1' 'TEL_LPS' 'TEL_TNF' 'TEL_IL-6' 'TEL_Mix']
drugList: ['DMSO' 'CIM' 'RAN' 'LEV' 'TRO' 'BUS' 'NEF' 'ASP' 'NIM' 'CLA' 'TEL']
drugListToxic: [0 0 1 0 1 0 1 0 1 1 1]
drugs: (66, 11)
cytokineList: ['NoCyt' 'IL-1' 'LPS' 'TNF' 'IL-6' 'Mix']
ind4pProtein: [ 7  8  9 10 11 12 31 32 33 34 35 36 37 38 39 40 41 42 85 86 87 88 89 90]
```

(1) Perform PLSR on the matrixes X and Y. Plot the percent variance explained. How many principal components do you need for each to explain 90% of the Y variance? Discuss your findings.

Hint: Be sure you are normalizing each dataset as needed for the analysis.

In [6]:

```
# Note: PLSR scales the data by default
#X_norm = StandardScaler().fit_transform(X)
#Y_norm = StandardScaler().fit_transform(Y)

scores = []

for i in range(1, 10):
    plsr = PLSRegression(n_components=i).fit(X, Y[:, -1]) # average of the late time-points used for prediction
    scores.append(plsr.score(X, Y[:, -1]))


plt.scatter(range(1,10), scores)
plt.ylabel('Variance Explained (%)')
plt.xlabel('Components')
plt.plot([0, 11], [0.9, 0.9], '--', color='blue')
plt.ylim(0, 1)
plt.xlim(0, 11)
plt.xticks(np.arange(1, 11, 1))
plt.show()

```



We would need 4 components to explain 90% of the variance observed under the average of the late time-points 4–48 hrs. We observe that the first component explains almost 68% of variance in Y. The consecutive PCs explain less and less variability eventually approaching 1.

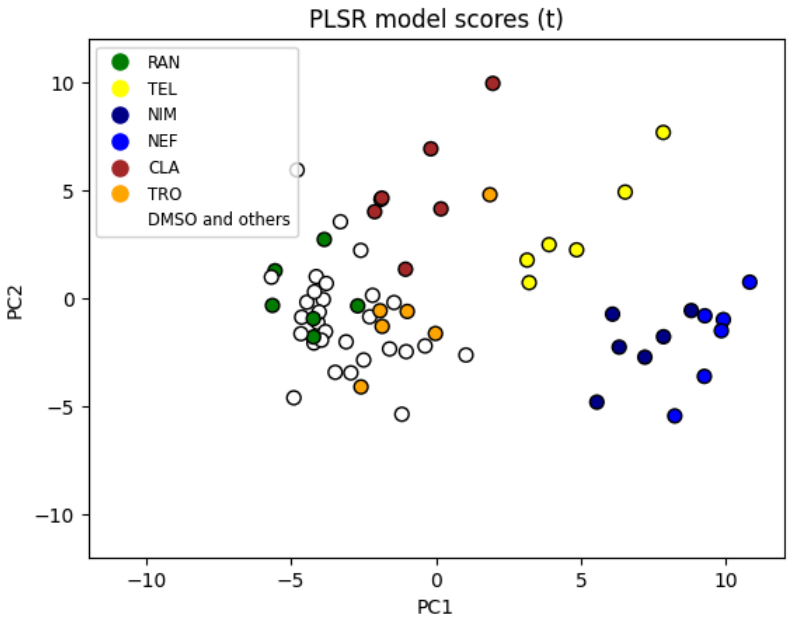
(2) How would you expect the percent of X variance explained to compare between PLSR and PCA? Why?

The percent of X variance explained in PCR would be higher compared to PLSR. PLSR focuses on maximixing covariance between X and Y, whereas PCA focuses on maximizing variance in X. Therefore 4 components are more than enough tto explain 90% of the covariance in the dataset. PCA would effectively explain the maximum variability observed in X, whereas PLSR would not.

(3) Recreate the Figure S2A plot from Cosgrove et al. supplementary information. This is the PLSR scores plot (for PC1 and PC2), with toxic drugs colored according to the drug type and all other drugs are not colored. Use the drugList, drugListToxic to identify these categories.

In [7]:

```
plsr = PLSRegression(n_components=4).fit(X, Y[:, -1])
scores = plsr.x_scores_
pcs = scores[:, :2]

drug_colors = {
    'RAN': 'green',
    'TEL': 'yellow',
    'NIM': 'darkblue',
    'NEF': 'blue',
    'CLA': 'brown',
    'TRO': 'orange',
    'DMSO and others': 'white'
}

colors = [
    drug_colors[drug] if (np.any(drugs[i] == 1) and (drug := drugList[np.argwhere(drugs[i] == 1)][0][0]) in drug_co
    else 'white'
    for i in range(len(scores))
]

# Scatter plot with legend
plt.scatter(x=pcs[:, 0], y=pcs[:, 1], c=colors, s=50, edgecolor='k')
plt.xlim(-12, 12)
plt.ylim(-12, 12)
plt.legend(handles=[plt.Line2D([0], [0], marker='o', color='w', markerfacecolor=color, markersize=10, label=drug) f
           loc='upper left',
           fontsize='small',
           bbox_to_anchor=(0, 1))

# Labels and title
plt.xlabel('PC1')
plt.ylabel('PC2')
plt.title('PLSR model scores (t)')
plt.show()
```

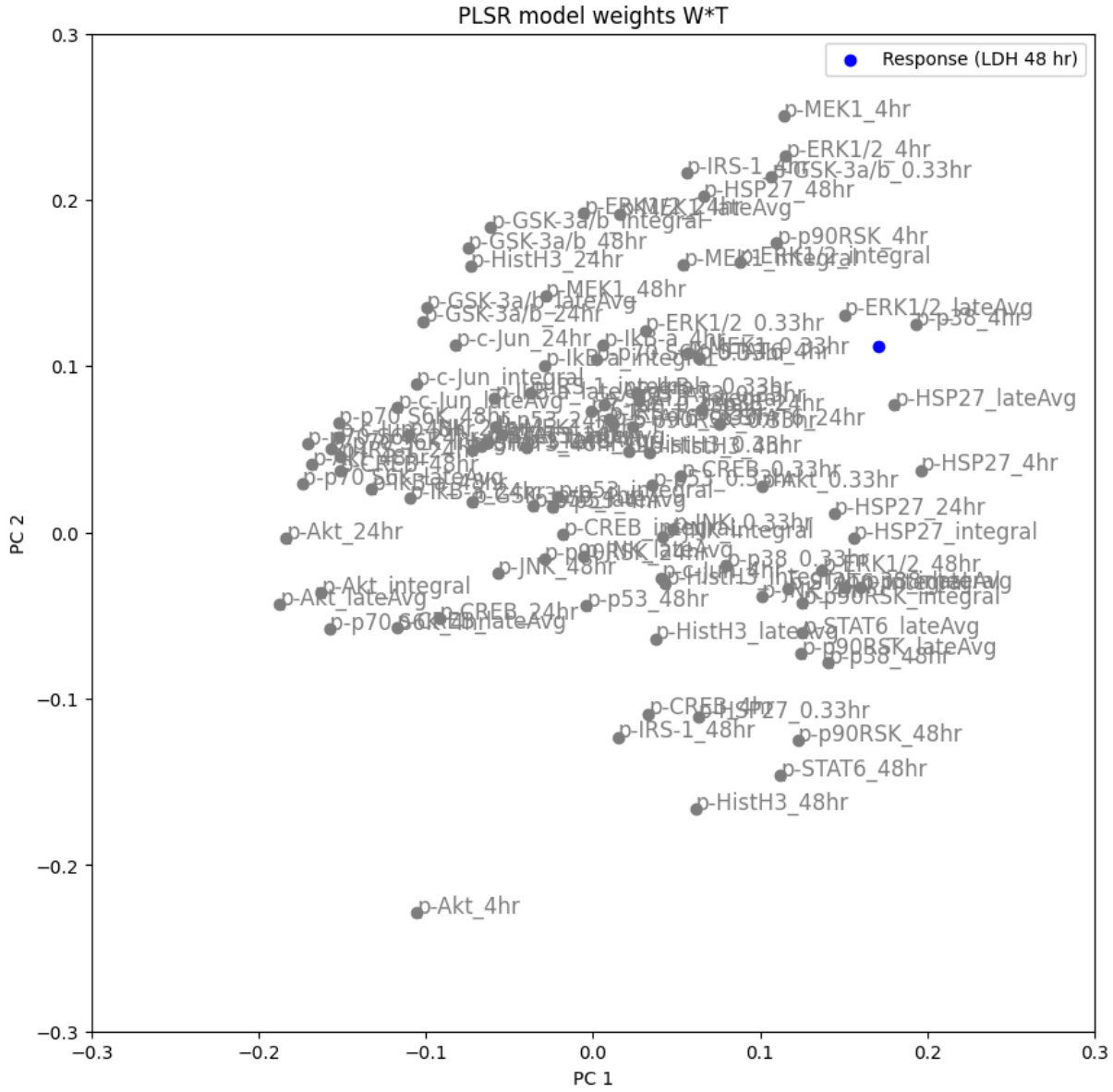


(4) Create the loadings plot corresponding to (3). Interpret the results shown on the plot.

In [8]:

```
weights = plsr.x_weights_[:, :2]
response = plsr.y_weights_

print(weights.shape, response.shape)

plt.figure(figsize=(10,10))
plt.scatter(weights[:, 0], weights[:, 1], color='grey')
plt.scatter(response[:, 0], response[:, 1], color='blue', label= 'Response (LDH 48 hr)')

for i, feature in enumerate(phosphoproteins):
    plt.text(weights[i, 0], weights[i, 1], feature, fontsize=12, color='grey')

# The six phosphoproteins (p-MEK1, p-ERK1/2, p-Akt, p-70 S6K, p-p38, p-HSP27) should surround the response variable

plt.legend()
plt.xlim(-0.3, 0.3)
plt.ylim(-0.3, 0.3)
plt.xlabel('PC 1')
plt.ylabel('PC 2')
plt.title('PLSR model weights W*T')
plt.show()

```

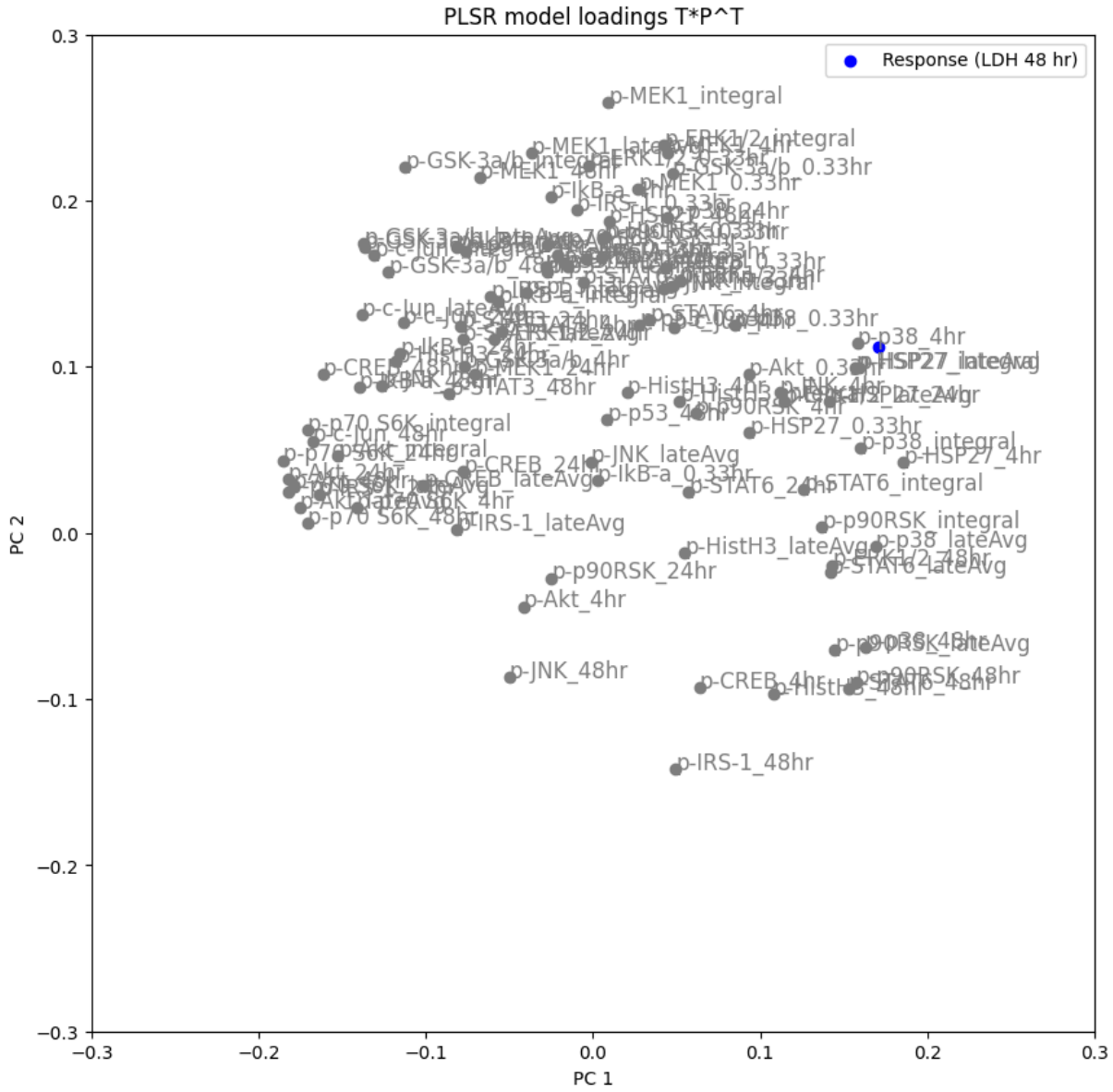
```
(102, 2) (1, 4)
```



In [9]:

```
loadings = plsr.x_loadings_[:, :2]
response = plsr.y_loadings_

print(loadings.shape, response.shape)

plt.figure(figsize=(10,10))
plt.scatter(loadings[:, 0], loadings[:, 1], color='grey')
plt.scatter(response[:, 0], response[:, 1], color='blue', label= 'Response (LDH 48 hr)')

for i, feature in enumerate(phosphoproteins):
    plt.text(loadings[i, 0], loadings[i, 1], feature, fontsize=12, color='grey')

plt.legend()
plt.xlim(-0.3, 0.3)
plt.ylim(-0.3, 0.3)
plt.xlabel('PC 1')
plt.ylabel('PC 2')
plt.title('PLSR model loadings T*P^T')
plt.show()
```

```
(102, 2) (1, 4)
```



Here, we display the weights plot above the loadings plot. Weights are used to compute the scores from the original componenet and they tell us the imporatnce of each varaible. The weights plot shows the directionality of each protein and its relationship to the scores plotted in the previous diagram, which depicts responses related to toxic and non-toxic drugs. The authors of the paper incorrectly claim that they are displaying the loadings plot; instead, they are plotting the x and y weights. Weights are useful for understanding the covariance structure and how the components are constructed to relate X and Y.

We use loadings to show how each variable (phosphoproteins and LDH release) contributes to each component in the PCA space. They can be thought of as correlations between the original data and the components.

The blue point highlighed represents our response variable Y. The other grey points represent the values of our signals across the first two principal components. The response is positively associated with PC1 and not PC2. The other phosphoproteins immediately surrounding it can also be good predictors of Y making them key candidates for further investigation in the context of LDH release regulation. This insight can guide further analysis or experimental validation to understand the biological role of these phosphoproteins in cell death response.

The weights (W) define the directions of the latent space that best predict Y. Weights are used to calculate scores.

The scores (T) represent the data in this latent space and are obtained by multiplying X onto with weights. Scores show how observations relate to components.

- $T = X * W$

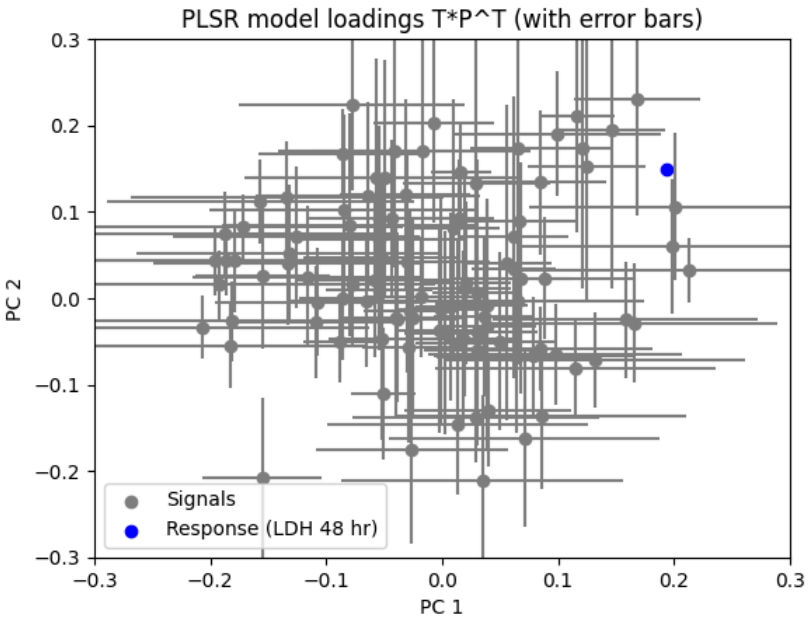
The loadings (P) allow reconstructing X from the scores, explaining how much variance of X is captured by each component. Loadings show how variables relate to components.

- X ≈ T * transpose(P)

(5) Add the variance of the loadings to your loadings plot (this can be shown as error bars). How does the variance of component one compare to that of component two? Would you expect a trend in the general variance versus component number?

In [13]:
```
from sklearn.utils import resample
import pandas as pd

loadings = plsr.x_weights_[:, :2]
response = plsr.y_weights_

print(loadings.shape, response.shape)


plt.scatter(loadings[:, 0], loadings[:, 1], color='grey', label = "Signals")
plt.scatter(response[:, 0], response[:, 1], color='blue', label= 'Response (LDH 48 hr)')
plt.legend()
plt.xlim(-0.3, 0.3)
plt.ylim(-0.3, 0.3)
plt.xlabel('PC 1')
plt.ylabel('PC 2')
plt.title('PLSR model loadings T*P^T (with error bars)')


output_data = []
n_bootstraps = 1000

for i in range(n_bootstraps):
    x_boot, y_boot = resample(X, Y[:,-1], replace=True)
    plsr = PLSRegression(n_components=2).fit(x_boot, y_boot)
    output_data.append({
        'X_predict' : plsr.predict(X),
        'X_loadings' : plsr.x_loadings_[:, :2],
        'Y_loadings' : plsr.y_loadings_,
        'Scores' : plsr.x_scores_
        }
)

output_df = pd.DataFrame(output_data)

vars = np.std(np.array(output_df['X_loadings']), axis=0)
plt.errorbar(loadings[:, 0], loadings[:, 1], xerr=vars[:, 0], yerr=vars[:, 1], linestyle='', color='grey')
plt.show()
```

```
(102, 2) (1, 2)
```



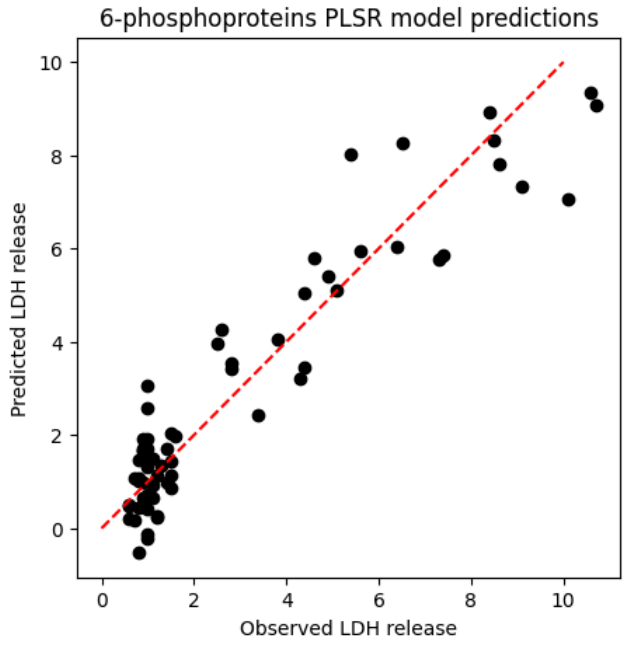
The variance of PC1 is generally higher compared to PC2. We should expect to see lower variance with increasing component numbers because the consecutive PCs are built with the residuals from the previous component. According to the paper, the positive model loadings imply signaling metrics with pro-death contributions, and negative model loadings imply pro-survival contributions. There is no direct correlation with variance, because we optimize for co-variance

(6) Recreate a 4-principal component model using PLSR with just the final 4 phosphoprotein model. Plot predicted v. observed LDH for this model. Report the model fitness (R^2). Define here how you are calculating R^2.

In [86]:

```
from sklearn.metrics import r2_score

'''
Metrics with variable importance score (VIP) score >1 have significant importance in the model whereas metrics with
significantly lack unique information in the model. Inspection of model loadings and VIP scores identified
four signaling pathways (MEK–ERK, Akt, p70 S6K, and p38–HSP27) with phosphoproteins having informative model
contributions from four or more signaling metrics.

5(A) An OPLSR model of LDH release was trained on the time-dependent signaling metrics from
six phosphoproteins (p-MEK1, p-ERK1/2, p-Akt, p-70 S6K, p-p38, p-HSP27) using data from donor #1.
This trained model was used to predict LDH release responses based on the same signaling metrics measured in hepatc
A correlation plot relating the observed and predicted LDH release responses for the both training data (donor #1)
which contained six treatments present and eight treatments not present in the training data. Test conditions (CHL
in the training data but nonetheless predicted accurately are noted. The model-predicted responses are presented as
mean prediction ± cross-validation standard error, calculated by jack-knifing. The test data are presented as mean
three biological replicates. Experimental and prediction uncertainties are not shown for the training data.

'''
print(ind4pProtein)
new_X = X[:, ind4pProtein]

plsr = PLSRegression(n_components=4)
plsr.fit(new_X, Y[:, -1])
predicted = plsr.predict(new_X)
score = plsr.score(new_X, Y[:, -1])

plt.figure(figsize=(5, 5))
plt.ylabel('Predicted LDH release')
plt.xlabel('Observed LDH release')
plt.title('6-phosphoproteins PLSR model predictions')

plt.plot([0, 10], [0, 10], '--r')
plt.scatter(Y[:, -1], predicted, color='k')
plt.show()

print(f'Model Fiting (R-squared): {score}')
```

```
[ 7  8  9 10 11 12 31 32 33 34 35 36 37 38 39 40 41 42 85 86 87 88 89 90]

/opt/homebrew/lib/python3.11/site-packages/IPython/core/pylabtools.py:152: MatplotlibDeprecationWarning: savefig() got
unexpected keyword argument "orientation" which is no longer supported as of 3.3 and will become an error two minor re
leases later
  fig.canvas.print_figure(bytes_io, **kw)
/opt/homebrew/lib/python3.11/site-packages/IPython/core/pylabtools.py:152: MatplotlibDeprecationWarning: savefig() got
unexpected keyword argument "dpi" which is no longer supported as of 3.3 and will become an error two minor releases l
ater
  fig.canvas.print_figure(bytes_io, **kw)
/opt/homebrew/lib/python3.11/site-packages/IPython/core/pylabtools.py:152: MatplotlibDeprecationWarning: savefig() got
unexpected keyword argument "facecolor" which is no longer supported as of 3.3 and will become an error two minor rele
ases later
  fig.canvas.print_figure(bytes_io, **kw)
/opt/homebrew/lib/python3.11/site-packages/IPython/core/pylabtools.py:152: MatplotlibDeprecationWarning: savefig() got
unexpected keyword argument "edgecolor" which is no longer supported as of 3.3 and will become an error two minor rele
ases later
  fig.canvas.print_figure(bytes_io, **kw)
/opt/homebrew/lib/python3.11/site-packages/IPython/core/pylabtools.py:152: MatplotlibDeprecationWarning: savefig() got
unexpected keyword argument "bbox_inches_restore" which is no longer supported as of 3.3 and will become an error two
minor releases later
  fig.canvas.print_figure(bytes_io, **kw)
```




```
Model Fiting (R-squared): 0.8821898904528919
```

R-squared was calcualted by fitting our PLSR model to X and Y matrices. Mathematically, it is 1 minus the sum of squares residuals divided by the total sum of squares. The model predicts Y from X. We calculate R^2 value with sklearn's r2_score function which uses the predicted and measured outcomes to inform about the goodness of fit.

(7) Cosgrove *et al* discusses their method for model validation using leave-one-out cross-validation. Calculate all LDH predictions for leave-one-out cross-validation and calculate the R^2 value for the resulting yfit values. What is the R^2 value? Why do you think it's important to perform cross-validation?

In [90]:
```
from sklearn.model_selection import cross_val_predict
predicted = cross_val_predict(plsr, X, Y[:, -1], cv=LeaveOneOut())
print(f'Model Fiting (R-squared): {r2_score(Y[:, -1], predicted)}')
```

```
Model Fiting (R-squared): 0.847813385573984
```

The R^2 value tells us about the goodness of fit. Here, our R^2 value has decreased from 0.88 to 0.85 after leave-one-out-CV. It is important to cross-validate data in order to see how well our model is able to generalize. In our previous approach, we included the testing set in the training process leading to an overestimation by the model. In CV, we aviod this by splitting our data into test and traning sets which enables an unbainsed training process.

(8) Now, instead of performing LOOCV, let's perform leave-one-cytokine-out cross-validation. That is, one cytokine at a time, leave out all the data for the `NoCyt`, `IL-1`, `LPS`, `TNF`, `IL-6`, or `Mix` conditions.

Hint: Look at `sklearn.model_selection.LeaveOneGroupOut`.

How does this affect your cross-validation? How do the two approaches here differ? When might each be most appropriate?

In [92]:
```
from sklearn.model_selection import LeaveOneGroupOut

print(cytokineList)
print(conditions)

groups = [next(i for i, cytokine in enumerate(cytokineList) if cytokine in cond) for cond in conditions]
predicted = cross_val_predict(plsr, X, Y[:, -1], cv=LeaveOneGroupOut().split(X, Y[:, -1], groups=groups))
print(f'Model Fiting (R-squared): {r2_score(Y[:, -1], predicted)}')
```

```
['NoCyt' 'IL-1' 'LPS' 'TNF' 'IL-6' 'Mix']
['DMSO_NoCyt' 'DMSO_IL-1' 'DMSO_LPS' 'DMSO_TNF' 'DMSO_IL-6' 'DMSO_Mix'
 'CIM_NoCyt' 'CIM_IL-1' 'CIM_LPS' 'CIM_TNF' 'CIM_IL-6' 'CIM_Mix'
 'RAN_NoCyt' 'RAN_IL-1' 'RAN_LPS' 'RAN_TNF' 'RAN_IL-6' 'RAN_Mix'
 'LEV_NoCyt' 'LEV_IL-1' 'LEV_LPS' 'LEV_TNF' 'LEV_IL-6' 'LEV_Mix'
 'TRO_NoCyt' 'TRO_IL-1' 'TRO_LPS' 'TRO_TNF' 'TRO_IL-6' 'TRO_Mix'
 'BUS_NoCyt' 'BUS_IL-1' 'BUS_LPS' 'BUS_TNF' 'BUS_IL-6' 'BUS_Mix'
 'NEF_NoCyt' 'NEF_IL-1' 'NEF_LPS' 'NEF_TNF' 'NEF_IL-6' 'NEF_Mix'
 'ASP_NoCyt' 'ASP_IL-1' 'ASP_LPS' 'ASP_TNF' 'ASP_IL-6' 'ASP_Mix'
 'NIM_NoCyt' 'NIM_IL-1' 'NIM_LPS' 'NIM_TNF' 'NIM_IL-6' 'NIM_Mix'
 'CLA_NoCyt' 'CLA_IL-1' 'CLA_LPS' 'CLA_TNF' 'CLA_IL-6' 'CLA_Mix'
 'TEL_NoCyt' 'TEL_IL-1' 'TEL_LPS' 'TEL_TNF' 'TEL_IL-6' 'TEL_Mix']
Model Fiting (R-squared): 0.8624270850931157
```

The results show that our R^2 has slightly improved with leave-one-cytokine-out cross validation compared to the previous iteration with LOOCV. This shows that the models predictive power had increased slightly. LOGO CV here has helped us determine whether coupling a cytokine with a drug really helps model learn unique features compared to there is no cytokine coupled or vice versa. LOOCV might be useful when testing a models overall ability to generalize.