Fitting drug response curves with sigmoid function

In [1]:

```
import pandas as pd
import numpy as np
from tqdm import tqdm
import warnings
warnings.filterwarnings("ignore")
import os, sys
sys.path.insert(1, os.path.relpath("../functions"))
from fitting import *
from plotting import *


_FOLDER = "../data/"
_FOLDER_2 = "../figures/"
_FOLDER_3 = "../results/"
```

Fitting data

In [2]:

```
df = pd.read_csv(_FOLDER_3+"filt_123.csv")
conc_columns= ["fd_num_"+str(i) for i in range(10)]
response_norm = ['norm_cells_'+str(i) for i in range(10)]
df.columns
```

Out[2]:

```
Index(['CELL_LINE_NAME', 'COSMIC_ID', 'DRUG_ID', 'DRUGID_COSMICID',
       'FOLD_DILUTION', 'MAX_CONC', 'fd_num_0', 'fd_num_1', 'fd_num_2',
       'fd_num_3', 'fd_num_4', 'fd_num_5', 'fd_num_6', 'fd_num_7', 'fd_num_8',
       'fd_num_9', 'norm_cells_0', 'norm_cells_1', 'norm_cells_2',
       'norm_cells_3', 'norm_cells_4', 'norm_cells_5', 'norm_cells_6',
       'norm_cells_7', 'norm_cells_8', 'norm_cells_9', 'drug_name', 'CCL_name',
       'dif_first', 'dif_last'],
      dtype='object')
```

sigmoid_4_param

```
# sigmoid_4_param

y = 1/ (L + np.exp(-k*(x-x0))) + d)

x0 -  p - position, correlation with IC50 or EC50
L = 1 protect from devision by zero if x is too small
k = -1/s (s -shape parameter)
d - determines the vertical position of the sigmoid - shift on y axis
```

In [3]:

```
%%time
fitting_function = "sigmoid_4_param"
r2, fit_param = fitting_column(df, df.index, x_columns=conc_columns, y_columns= response_norm,
                               fitting_function = fitting_function, default_param=True)
df[fitting_function+"_r2"] = r2
df[fitting_function] = fit_param
df= df[df[fitting_function+"_r2"]>0]
print("R2>0:", df.shape)
print("R2>0.9", df[df[fitting_function+"_r2"]>0.9].shape[0])
print("Number of samples with fitting <0.1:", df[df[fitting_function+"_r2"]<0.1].shape[0])
print("")
```

```
100%|██████████| 2776/2776 [00:08<00:00, 340.49it/s]

<function sigmoid_4_param at 0x7f91002d5ae8>
R2>0: (2755, 32)
R2>0.9 2703
Number of samples with fitting <0.1: 21

CPU times: user 7.79 s, sys: 328 ms, total: 8.12 s
Wall time: 8.19 s
```

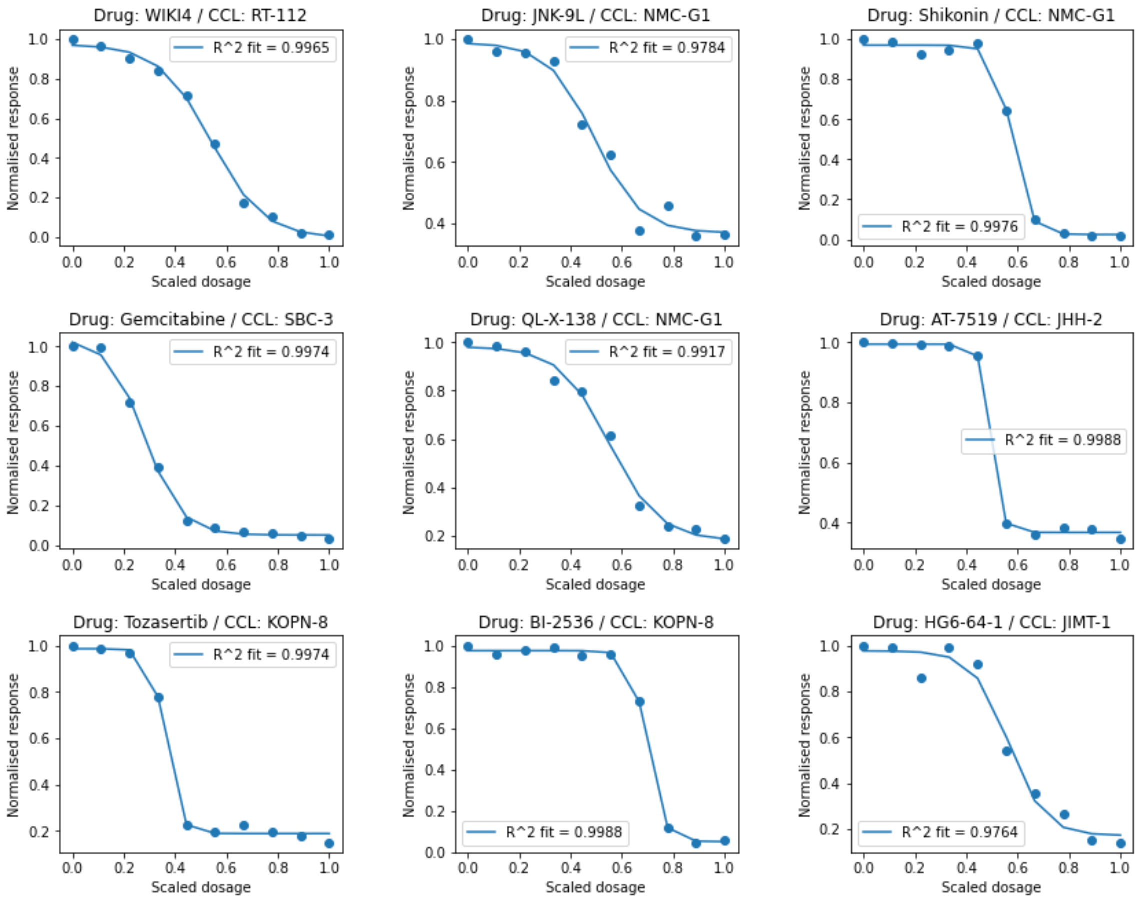
Visual Analysis of fitting efficiency

All samples

In [4]:

```
# ShowResponseCurvesWithFitting(df.drop(["drug_name", "CCL_name"],axis=1), plots_in_row=3, plots_
#                         indexes=df.index[:9],fitting_function = fitting_function,
#                                   fitting_parameters =fitting_function)

show_response_curves_with_fitting(df, plots_in_row=3, plots_in_column=3, x_columns=conc_columns,
                        indexes=df.index[:9],fitting_function = fitting_function,
                                  fitting_parameters =fitting_function)
```

```
Figures titles: Index_DRUG_ID_COSMIC_ID (COSMIC_ID is a cell line)
```

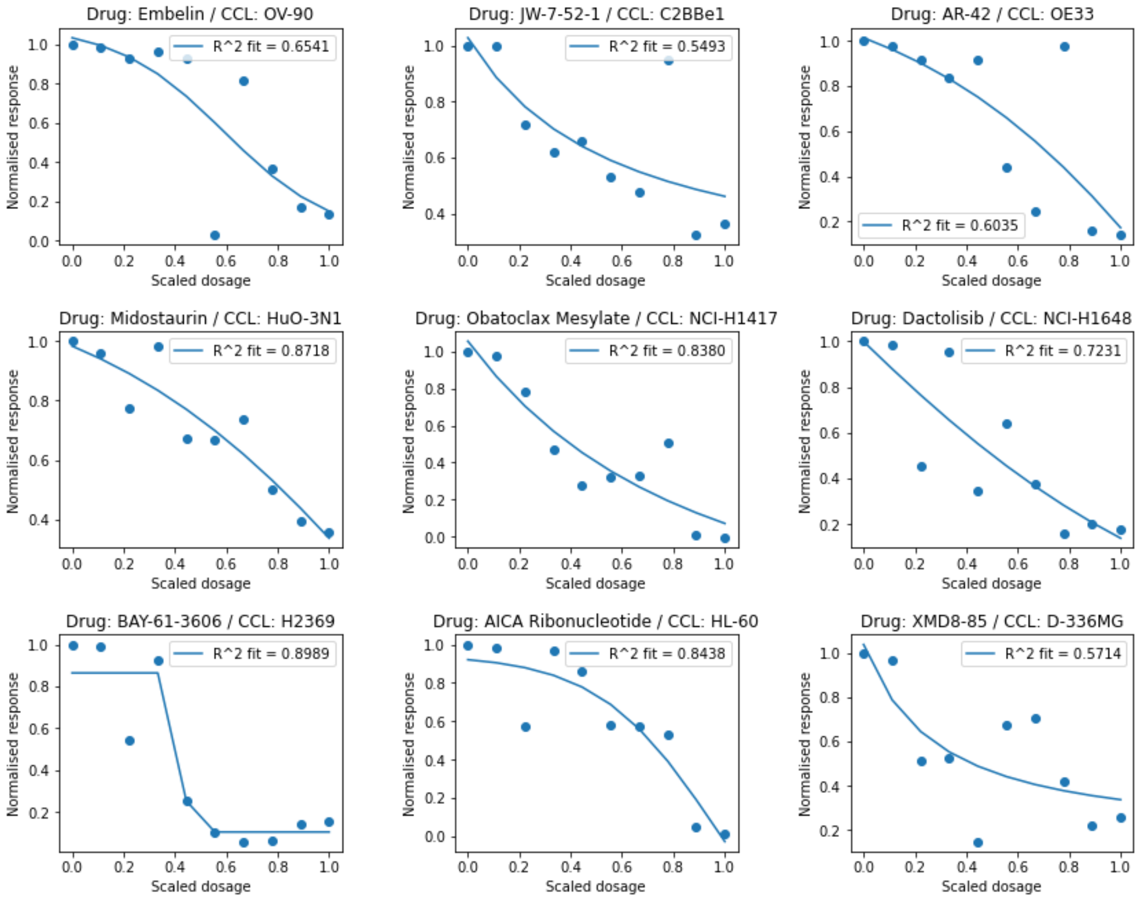


Bad fitting examples (examination after analysis of predictive models)

In [5]:

```
df2= df[(df[fitting_function+"_r2"]>0.1)& (df[fitting_function+"_r2"]<0.9)]

show_response_curves_with_fitting(df2, plots_in_row=3, plots_in_column=3, x_columns=conc_columns
                                 indexes= df2.index[:9],fitting_function = fitting_function,
                                               fitting_parameters =fitting_function)
```

```
Figures titles: Index_DRUG_ID_COSMIC_ID (COSMIC_ID is a cell line)
```

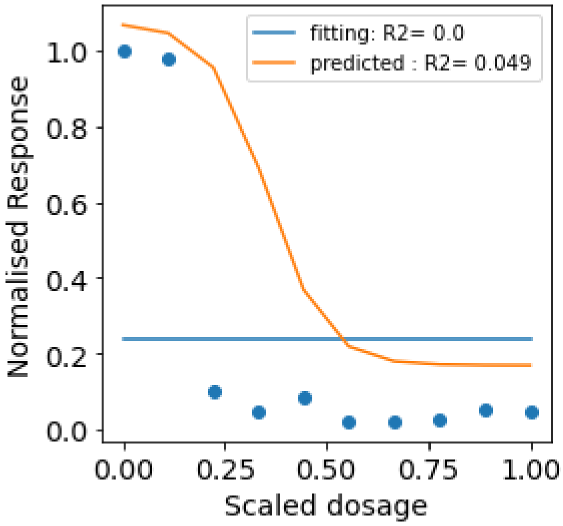


Outliers in predictive models

In [6]:

```
ind = int(df[(df["DRUG_ID"]==180)& (df["COSMIC_ID"]==907064)].index[0])
fitting_parameters = fitting_function
predicted_param = [0.348604, 1.106316, -14.202945, 0.168828]
save_fig_name = _FOLDER_2+"outlier_coef1_1.png"

show_one_fitting(df, ind, conc_columns, response_norm, fitting_function, fitting_parameters, pred
```

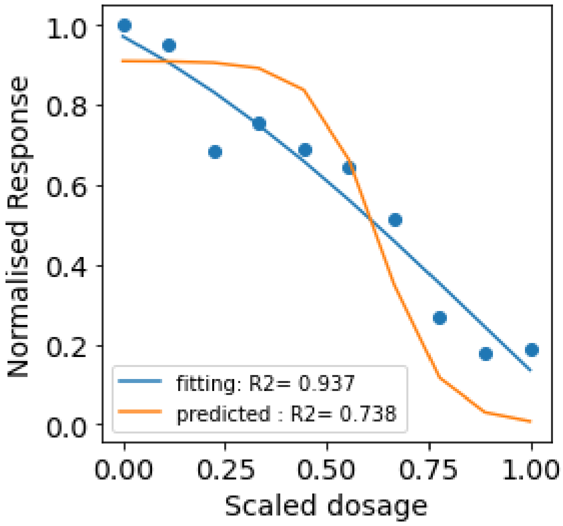
```
Fitting parameters: [  9.5763911   -5.03279099 -11.74360689   0.43769208]
Predicted parameters: [0.348604, 1.106316, -14.202945, 0.168828]
```



In [7]:

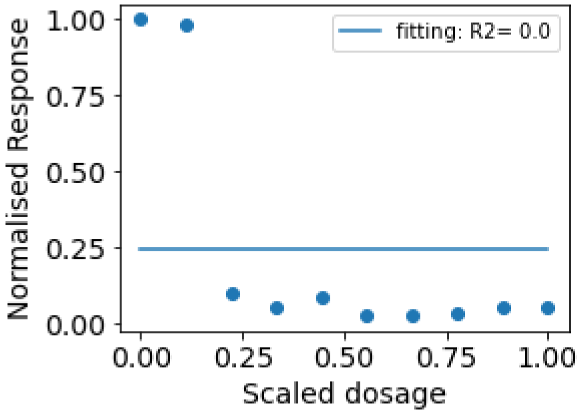
```
ind =int(df[(df["DRUG_ID"]==173)& (df["COSMIC_ID"]==687777)].index[0])
fitting_parameters =fitting_function
predicted_param = [0.623563, 1.099364, -13.124646, -8.772640e-15]
save_fig_name = _FOLDER_2+"outlier_coef3.png"

show_one_fitting(df, ind, conc_columns, response_norm, fitting_function, fitting_parameters, pred
```

```
Fitting parameters: [ 1.2913904   0.4645758  -1.82168875 -0.81502011]
Predicted parameters: [0.623563, 1.099364, -13.124646, -8.77264e-15]
```



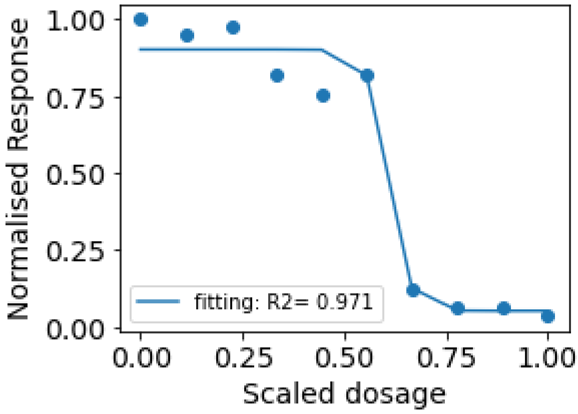
In [8]:

```
ind =int(df[(df["DRUG_ID"]==180)& (df["COSMIC_ID"]==907064)].index[0])
fitting_parameters =fitting_function
save_fig_name = _FOLDER_2+"outlier_coef1_2.png"

fig_size = (4,3)
show_one_fitting(df, ind, conc_columns, response_norm, fitting_function, fitting_parameters, fig_
```



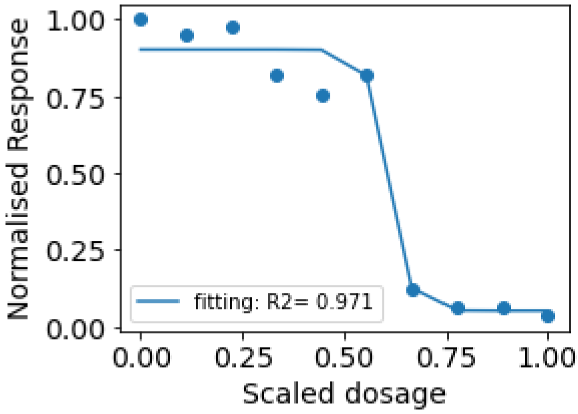
In [9]:

```
ind =int(df[(df["DRUG_ID"]==273)& (df["COSMIC_ID"]==907071)].index[0])
fitting_parameters =fitting_function
save_fig_name = _FOLDER_2+"filt_fit_0.png"

fig_size = (4,3)
show_one_fitting(df, ind, conc_columns, response_norm, fitting_function, fitting_parameters, fig_
```



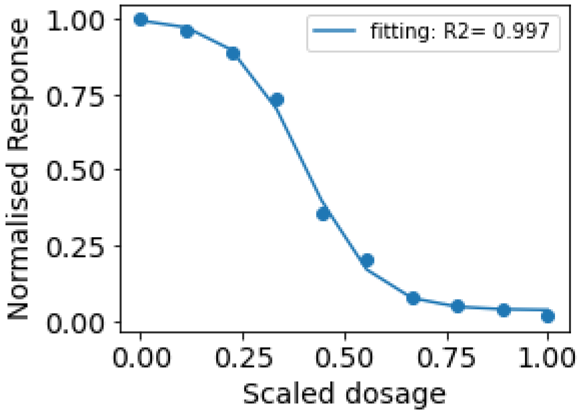
In [10]:

```
ind =int(df[(df["DRUG_ID"]==273)& (df["COSMIC_ID"]==907071)].index[0])
fitting_parameters = fitting_function
save_fig_name = _FOLDER_2+"filt_fit_0.png"

fig_size = (4,3)
show_one_fitting(df, ind, conc_columns, response_norm, fitting_function, fitting_parameters, fig_
```



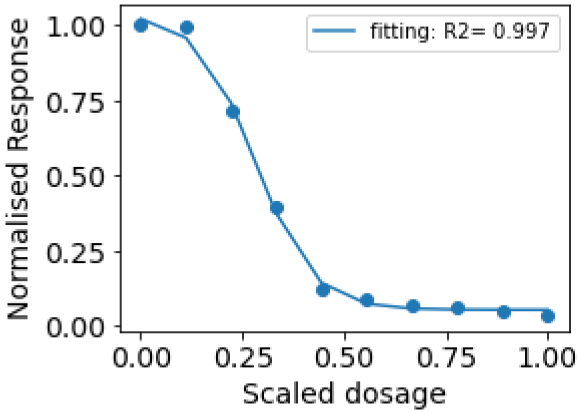
In [11]:

```
ind =int(df[(df["DRUG_ID"]==274)& (df["COSMIC_ID"]==1240223)].index[0])
save_fig_name = _FOLDER_2+"filt_fit_1.png"

fig_size = (4,3)
show_one_fitting(df, ind, conc_columns, response_norm, fitting_function, fitting_parameters, fig_
```



In [12]:

```
ind =int(df[(df["DRUG_ID"]==135)& (df["COSMIC_ID"]==753610)].index[0])
save_fig_name = _FOLDER_2+"filt_fit_2.png"

fig_size = (4,3)
show_one_fitting(df, ind, conc_columns, response_norm, fitting_function, fitting_parameters, fig_
```



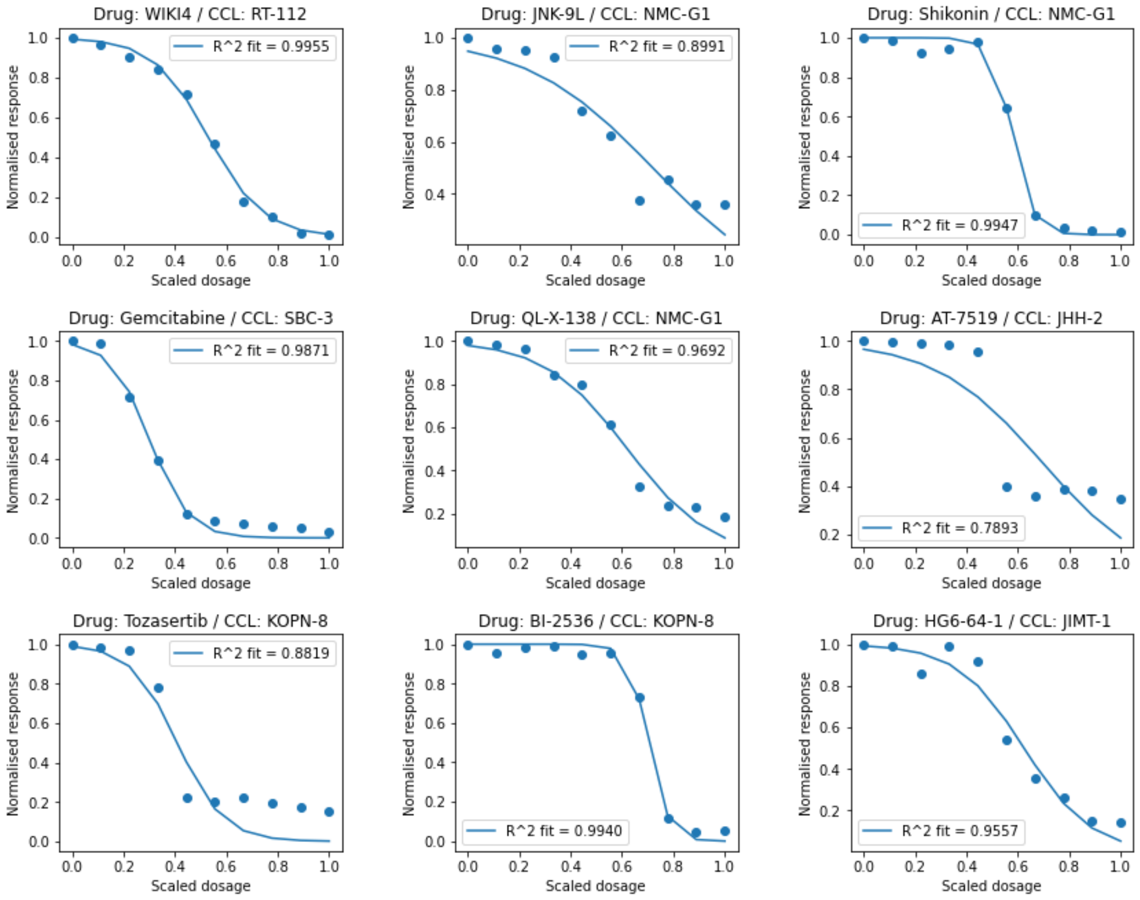
sigmoid_2_param

```
#sigmoid_2_param:

y = 1.0 / (1.0 + np.exp((x-p)/s))

x - dosage [0, 1],
p - position,        default=0.4
s - shape parameter,  default=-1
```

```
In [13]:
```

```
%%time
fitting_function = "sigmoid_2_param"
# "sigmoid_Wang" we don't need default_param_number
r2, fit_param = fitting_column(df, df.index, x_columns=conc_columns, y_columns= response_norm,
                               fitting_function = fitting_function, default_param=True)
df[fitting_function+"_r2"] = r2
df[fitting_function] = fit_param
df= df[df[fitting_function+"_r2"]!=0]
print("R2>0:", df.shape)
print("R2>0.9", df[df[fitting_function+"_r2"]>0.9].shape[0])
show_response_curves_with_fitting(df, plots_in_row=3, plots_in_column=3, x_columns=conc_columns,
                        indexes=df.index[:9],fitting_function = fitting_function,
                                  fitting_parameters =fitting_function)
```

```
100%|██████████| 2755/2755 [00:05<00:00, 539.10it/s]

<function sigmoid_2_param at 0x7f91002d5730>
R2>0: (2755, 34)
R2>0.9 2457
Figures titles: Index_DRUG_ID_COSMIC_ID (COSMIC_ID is a cell line)
CPU times: user 4.86 s, sys: 197 ms, total: 5.05 s
Wall time: 5.4 s
```



sigmoid_3_param

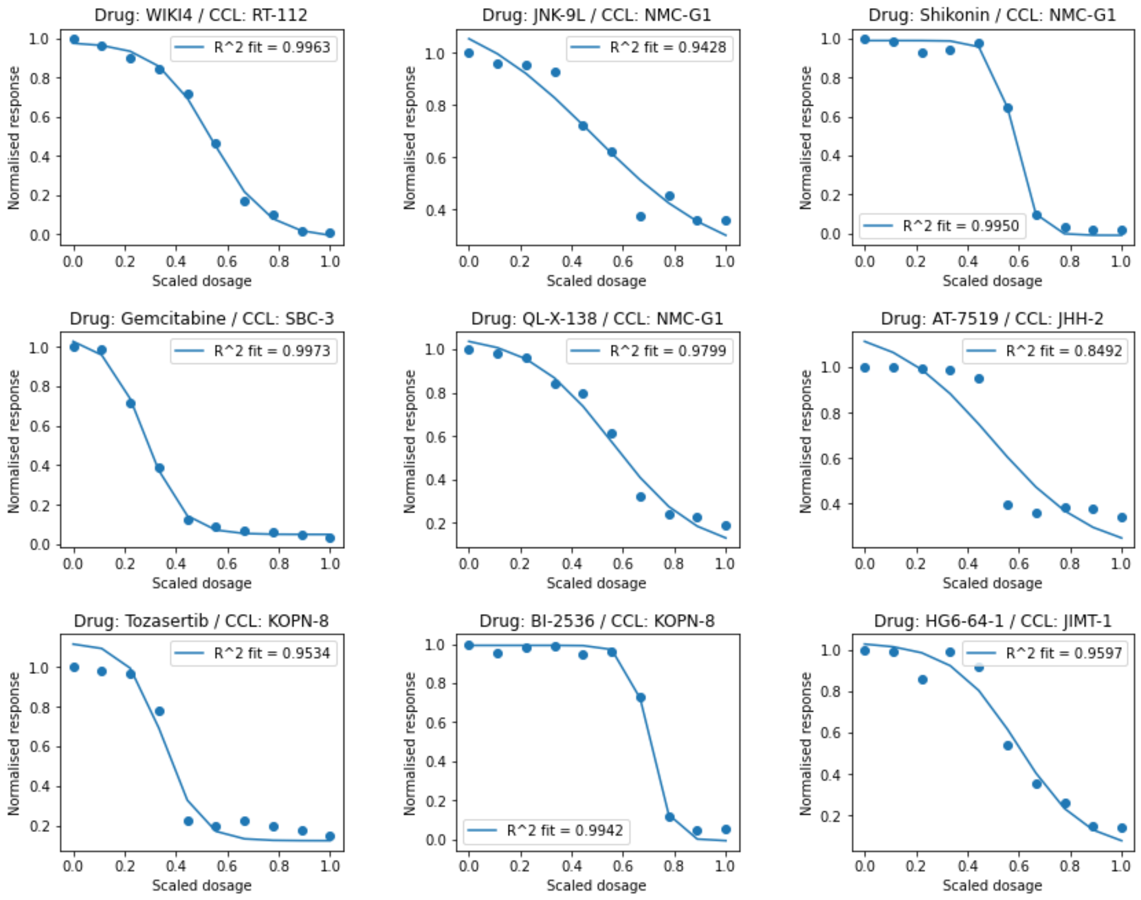
```
# sigmoid_3_param
y = 1/ (1 + np.exp(-k*(x-x0))) + d

x0 -  p - position, correlation with IC50 or EC50
k = -1/s (s -shape parameter)
```

```
d - determines the vertical position of the sigmoid - shift on y axis - better fitting then
Dennis Wang's sigmoid
```

In [14]:

```
%%time
fitting_function = "sigmoid_3_param"

r2, fit_param = fitting_column(df, df.index, x_columns=conc_columns, y_columns= response_norm,
                               fitting_function = fitting_function, default_param=True)
df[fitting_function+"_r2"] = r2
df[fitting_function] = fit_param
df= df[df[fitting_function+"_r2"]!=0]

print("R2>0:", df.shape)
print("R2>0.9", df[df[fitting_function+"_r2"]>0.9].shape[0])
show_response_curves_with_fitting(df, plots_in_row=3, plots_in_column=3, x_columns=conc_columns,
                        indexes=df.index[:9],fitting_function = fitting_function,
                                  fitting_parameters =fitting_function)
```

```
100%|██████████| 2755/2755 [00:11<00:00, 233.67it/s]

<function sigmoid_3_param at 0x7f91002d5a60>
R2>0: (2755, 36)
R2>0.9 2683
Figures titles: Index_DRUG_ID_COSMIC_ID (COSMIC_ID is a cell line)
CPU times: user 9.46 s, sys: 417 ms, total: 9.88 s
Wall time: 12.1 s
```



fsigmoid

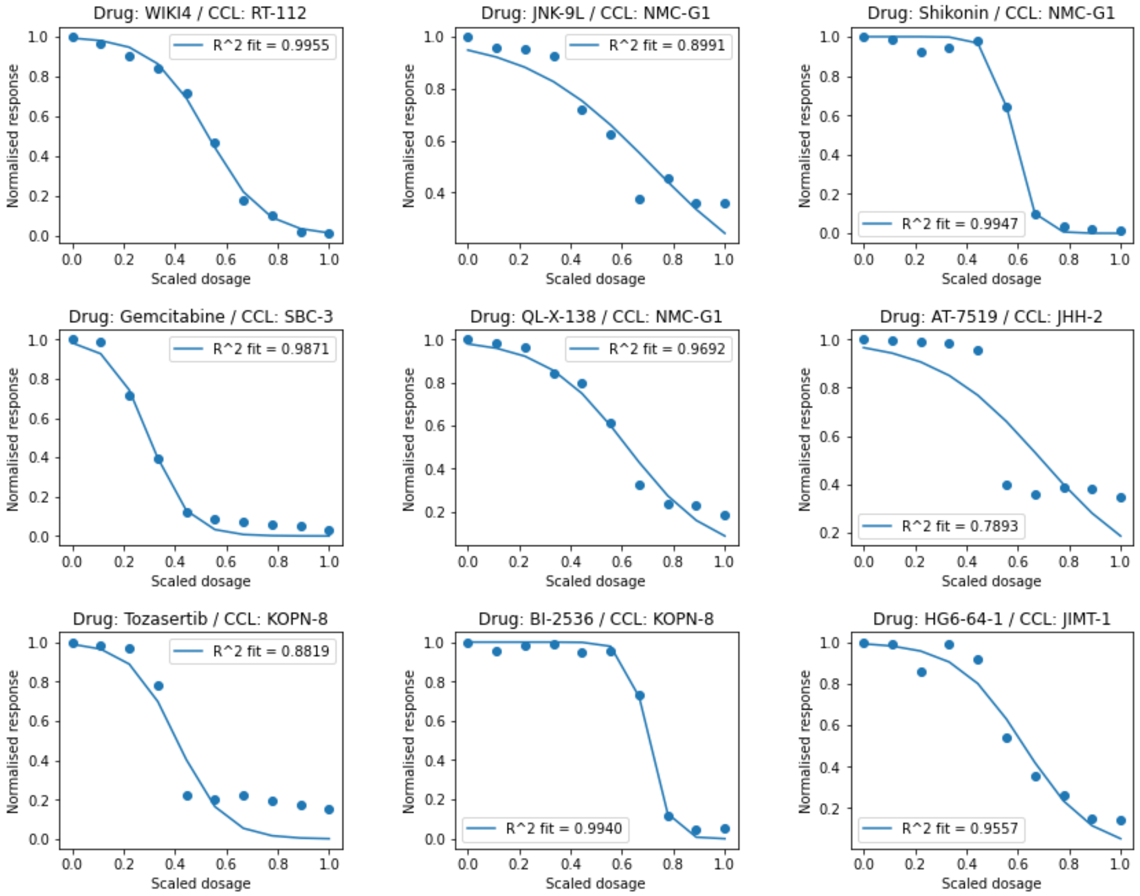
```
# fsigmoid
```

```
y =1.0 / (1.0 + np.exp(-k*(x-p))

    x = x  - dosage [0, 1]
    p - position [0,1],           default=0.4
    k = -1/s (s -shape parameter) default=0.4
```

In [15]:

```
# %%time
fitting_function = "fsigmoid"

r2, fit_param = fitting_column(df, df.index, x_columns=conc_columns, y_columns= response_norm,
                               fitting_function = fitting_function, default_param=True)
df[fitting_function+"_r2"] = r2
df[fitting_function] = fit_param
df= df[df[fitting_function+"_r2"]!=0]
print(df.shape)

show_response_curves_with_fitting(df, plots_in_row=3, plots_in_column=3, x_columns=conc_columns,
                        indexes=df.index[:9],fitting_function = fitting_function,
                                  fitting_parameters =fitting_function)
```

```
100%|██████████| 2755/2755 [00:05<00:00, 494.88it/s]

<function fsigmoid at 0x7f91002d59d8>
(2755, 38)
Figures titles: Index_DRUG_ID_COSMIC_ID (COSMIC_ID is a cell line)
```



logistic_4_param

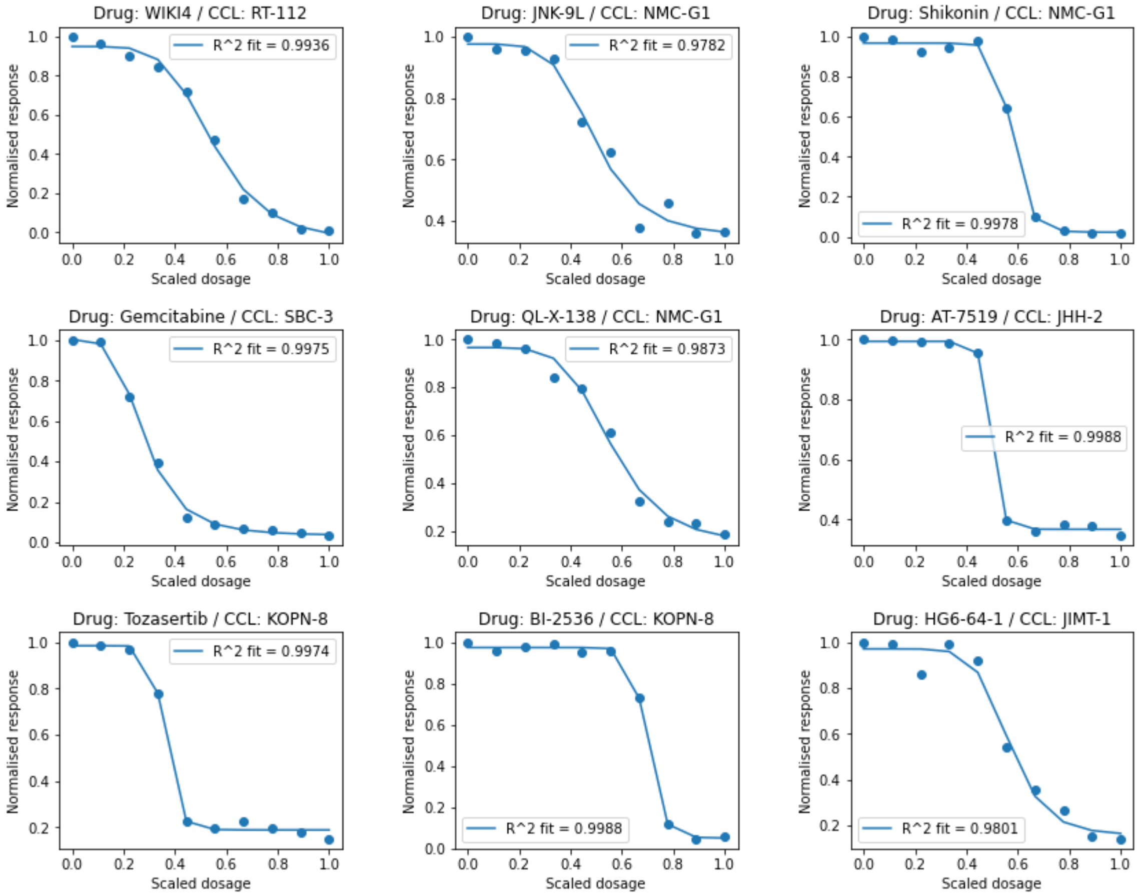
```
# logistic_4_param
```

```
y =(A-d)/(1.0+((x/C)**B)) + d

    (A - d) = 1 in sigmoid_2_param:
    (x/C)**B  - corresponds to np.exp((x-p)/s
    d - determines the vertical position of the graph
```

In [16]:

```
%%time
fitting_function = "logistic_4_param"
r2, fit_param = fitting_column(df, df.index, x_columns=conc_columns, y_columns= response_norm,
                               fitting_function = fitting_function, default_param=True)
df[fitting_function+"_r2"] = r2
df[fitting_function] = fit_param
df= df[df[fitting_function+"_r2"]!=0]

print("R2>0:", df.shape)
print("R2>0.9", df[df[fitting_function+"_r2"]>0.9].shape[0])
show_response_curves_with_fitting(df, plots_in_row=3, plots_in_column=3, x_columns=conc_columns,
                        indexes=df.index[:9],fitting_function = fitting_function,
                              fitting_parameters =fitting_function)
```

```
100%|██████████| 2755/2755 [00:11<00:00, 242.35it/s]

<function logistic_4_param at 0x7f91002d5c80>
R2>0: (2755, 40)
R2>0.9 2720
Figures titles: Index_DRUG_ID_COSMIC_ID (COSMIC_ID is a cell line)
CPU times: user 9.75 s, sys: 477 ms, total: 10.2 s
Wall time: 11.7 s
```



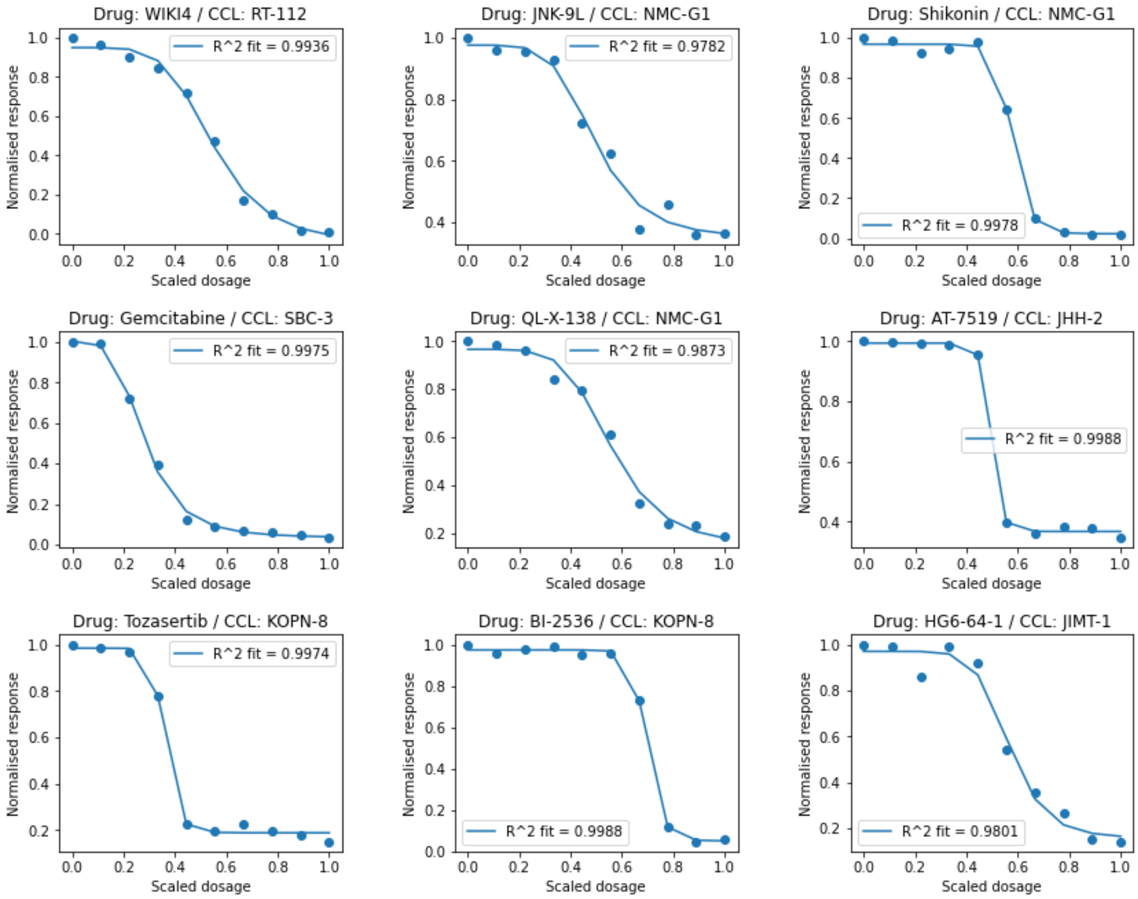
LL4_4_param

```
# ll4_4_param

y= (c-d)/(1 + np.exp( b*(np.log(x)-np.log(e) ))) + d

     - b: hill slope
     - d: min response - determines the vertical position of the graph
     - c: max response
     - e: EC50
     c-d - difference between max and min responses
     np.exp( b* (np.log(x)-np.log(e)) -  np.exp((x-p)/s in sigmoid_2_param
     b- hill slope = 1/s - shape parameter
     np.log(x)-np.log(e) == x-p in sigmoid_2_param
```

In [17]:

```
%%time
fitting_function = "ll4_4_param"
r2, fit_param = fitting_column(df, df.index, x_columns=conc_columns, y_columns= response_norm,
                               fitting_function = fitting_function, default_param=True)
df[fitting_function+"_r2"] = r2
df[fitting_function] = fit_param
df= df[df[fitting_function+"_r2"]!=0]
print("R2>0:", df.shape)
print("R2>0.9", df[df[fitting_function+"_r2"]>0.9].shape[0])
show_response_curves_with_fitting(df, plots_in_row=3, plots_in_column=3, x_columns=conc_columns,
                        indexes=df.index[:9],fitting_function = fitting_function,
                              fitting_parameters =fitting_function)
```

```
100%|██████████| 2755/2755 [00:13<00:00, 210.24it/s]

<function ll4_4_param at 0x7f91002d5b70>
R2>0: (2755, 42)
R2>0.9 2721
Figures titles: Index_DRUG_ID_COSMIC_ID (COSMIC_ID is a cell line)
CPU times: user 11.4 s, sys: 515 ms, total: 11.9 s
Wall time: 13.5 s
```



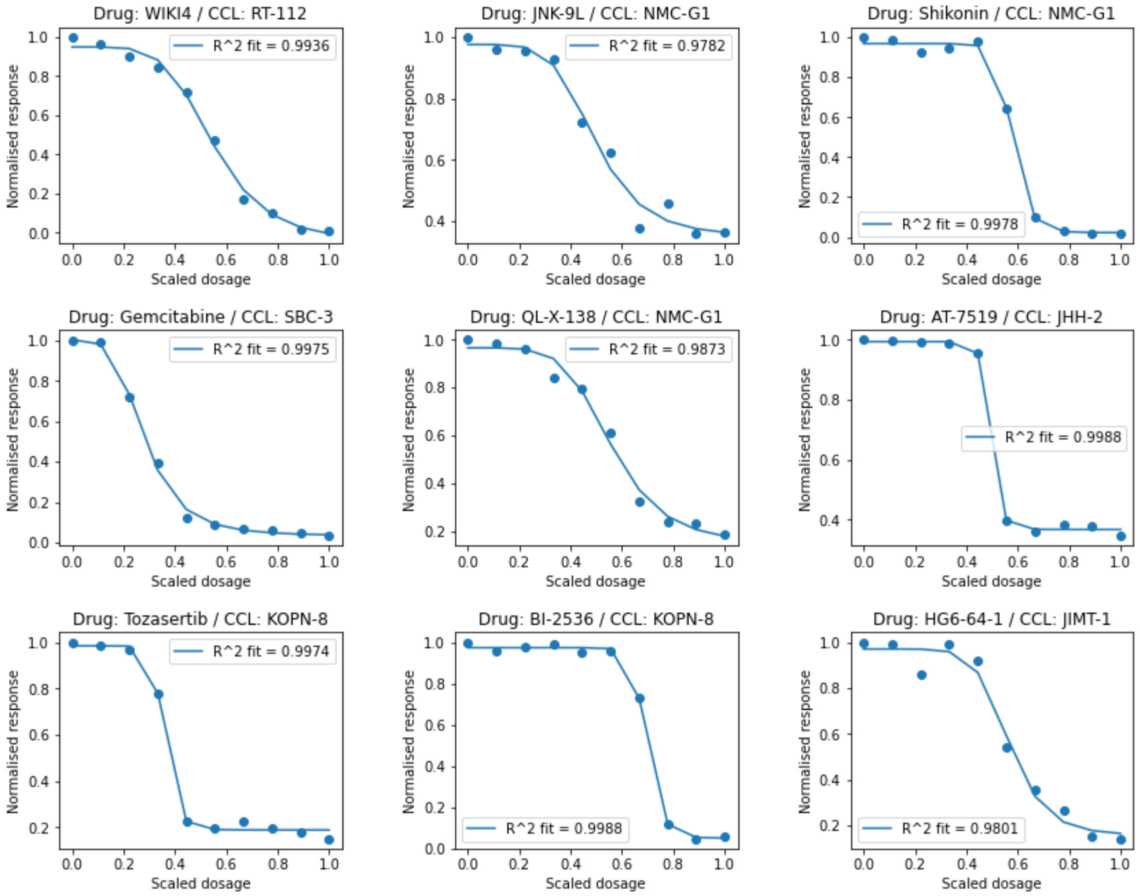
ll4R_4_param

```
#ll4R_4_param
y=(a-d)/(1+np.exp(b*np.log(x)- c)) + d
    a-d - difference between max and min responses
    np.exp( b* np.log(x) - e) -  np.exp((x-p)/s in sigmoid_2_param
    b - hill slope = 1/s - shape parameter
    np.log(x)- e/b == x-p in sigmoid_2_param
```

In [18]:

```
%%time
fitting_function = "ll4R_4_param"
r2, fit_param = fitting_column(df, df.index, x_columns=conc_columns, y_columns= response_norm,
                               fitting_function = fitting_function, default_param=True)
df[fitting_function+"_r2"] = r2
df[fitting_function] = fit_param
df= df[df[fitting_function+"_r2"]!=0]

print("R2>0:", df.shape)
print("R2>0.9", df[df[fitting_function+"_r2"]>0.9].shape[0])
show_response_curves_with_fitting(df, plots_in_row=3, plots_in_column=3, x_columns=conc_columns,
                        indexes=df.index[:9],fitting_function = fitting_function,
                                  fitting_parameters =fitting_function)
```

```
100%|██████████| 2755/2755 [00:13<00:00, 205.34it/s]

<function ll4R_4_param at 0x7f91002d5bf8>
R2>0: (2755, 44)
R2>0.9 2716
Figures titles: Index_DRUG_ID_COSMIC_ID (COSMIC_ID is a cell line)
CPU times: user 11.4 s, sys: 529 ms, total: 12 s
Wall time: 13.9 s
```



logLogist_3_param

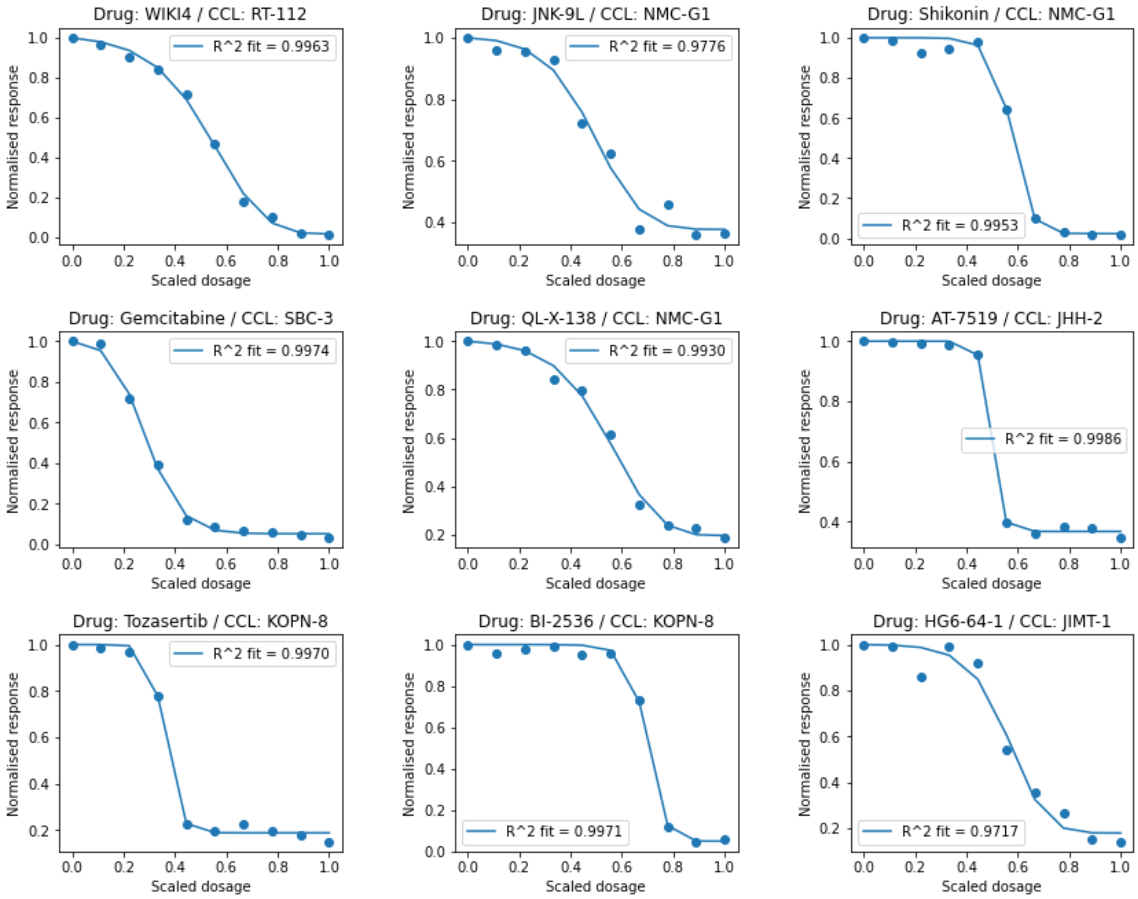
```
# logLogist_3_param
y= (1-E_inf)/(1+(np.log10(x)/EC50)**HS) + E_inf
```

```
    E = E_inf + (1 - E_inf)/(1 + (x/EC50)^HS)
    sigmoid_2_param: 1.0 / (1.0 + np.exp((x-p)/s)

    (A - d) = 1 in sigmoid_2_param:
    (np.log10(x)/EC50)**HS  - (in logistic4 (x/C)**B) corresponds to np.exp((x-p)/s

    E_inf - determines the vertical position of the graph /coefficient d, min response in other
functions
```

In [19]:

```
%%time
fitting_function = "logLogist_3_param"
r2, fit_param = fitting_column(df, df.index, x_columns=conc_columns, y_columns= response_norm,
                              fitting_function = fitting_function, default_param=True)
df[fitting_function+"_r2"] = r2
df[fitting_function] = fit_param
df= df[df[fitting_function+"_r2"]!=0]

print("R2>0:", df.shape)
print("R2>0.9", df[df[fitting_function+"_r2"]>0.9].shape[0])
show_response_curves_with_fitting(df, plots_in_row=3, plots_in_column=3, x_columns=conc_columns,
                          indexes=df.index[:9],fitting_function = fitting_function,
                                fitting_parameters =fitting_function)
```

```
100%|██████████| 2755/2755 [00:08<00:00, 324.21it/s]

<function logLogist_3_param at 0x7f91002d5d08>
R2>0: (2755, 46)
R2>0.9 2714
Figures titles: Index_DRUG_ID_COSMIC_ID (COSMIC_ID is a cell line)
CPU times: user 8.56 s, sys: 334 ms, total: 8.89 s
Wall time: 8.88 s
```



Comparison of fitting models

In [20]:

```
functions = {"fsigmoid",
             "sigmoid_2_param",
             "sigmoid_3_param",
             "sigmoid_4_param",
             "logistic_4_param",
            "ll4_4_param",
            "ll4R_4_param",
            "logLogist_3_param"}

functions_dict= dict(list(enumerate(functions)))
r2_columns = [fitting_function+"_r2" for fitting_function in functions]
```

In [21]:

```
df["better_fitting"] = np.argmax(df[r2_columns].values, axis=1)
r2_col_res = r2_columns +["better_fitting"]
df["better_fitting"] = df["better_fitting"].map(functions_dict)
df[r2_col_res].head()
```

Out[21]:

	sigmoid_4_param_r2	sigmoid_2_param_r2	sigmoid_3_param_r2	ll4_4_param_r2	ll4R_4_pa
0	0.996467	0.995452	0.996302	0.993608	0.
1	0.978440	0.899079	0.942793	0.978230	0.
2	0.997584	0.994659	0.995039	0.997801	0.
3	0.997357	0.987070	0.997270	0.997515	0.
4	0.991678	0.969244	0.979949	0.987341	0.

In [22]:

```
df["better_fitting"].value_counts()
```

Out[22]:

```
sigmoid_4_param      1122
logLogist_3_param     800
logistic_4_param      296
ll4R_4_param          278
ll4_4_param           258
sigmoid_3_param         1
Name: better_fitting, dtype: int64
```

In [23]:

```
r2_limit = 0.98
fitted_samples = {}
for function in functions:
    fitted_samples[function] = df[df[function+"_r2"]> r2_limit].shape[0]

pd.DataFrame(fitted_samples.values(), index=fitted_samples.keys(), columns= ["fitted_samples R2>'
            .sort_values("fitted_samples R2>"+str(r2_limit), ascending=False)
```

Out[23]:

	fitted_samples R2>0.98
sigmoid_4_param	2306
ll4_4_param	2283
logistic_4_param	2282
ll4R_4_param	2278
logLogist_3_param	2255
sigmoid_3_param	1829
sigmoid_2_param	1395
fsigmoid	1395

In [24]:

```
r2_limit = 0.95
fitted_samples = {}
for function in functions:
    fitted_samples[function] = df[df[function+"_r2"]> r2_limit].shape[0]

pd.DataFrame(fitted_samples.values(), index=fitted_samples.keys(), columns= ["fitted_samples R2>'
            .sort_values("fitted_samples R2>"+str(r2_limit), ascending=False)
```

Out[24]:

	fitted_samples R2>0.95
ll4_4_param	2636
logistic_4_param	2636
ll4R_4_param	2629
sigmoid_4_param	2614
logLogist_3_param	2610
sigmoid_3_param	2454
sigmoid_2_param	2051
fsigmoid	2051

In [25]:

```
r2_limit = 0.9
fitted_samples = {}
for function in functions:
    fitted_samples[function] = df[df[function+"_r2"]> r2_limit].shape[0]

pd.DataFrame(fitted_samples.values(), index=fitted_samples.keys(), columns= ["fitted_samples R2>"
            .sort_values("fitted_samples R2>"+str(r2_limit), ascending=False)
```

Out[25]:

	fitted_samples R2>0.9
ll4_4_param	2721
logistic_4_param	2720
ll4R_4_param	2716
logLogist_3_param	2714
sigmoid_4_param	2703
sigmoid_3_param	2683
sigmoid_2_param	2457
fsigmoid	2457

In [27]:

```
df.to_csv(_FOLDER_3+"fit_123.csv", index=False)
```

In [ ]:

```
```