Supporting information

Bayesian optimization of glycopolymer structures for the interaction with cholera toxin B subunit

Masanori Nagao, Osuke Nakahara, Xincheng Zhou, Hikaru Matsumoto and Yoshiko Miura*

Department of Chemical Engineering, Kyushu University, 744 Motooka, Nishi-ku, Fukuoka 819-0395, Japan.
E-mail: miuray@chem-eng.kyushu-u.ac.jp

Table of Contents

1. Materials and Methods

N,*N*-Dimethylacrylamide (DMA, 99%) was purchased from Tokyo Chemical Industry (Tokyo, Japan). EosinY disodium salt (85%), cholera toxin B subunit (CTB) and monosialoganglioside GM1 from bovine brain were purchased from Sigma Aldrich (St. Louis, USA). Potassium hydroxide (KOH), ascorbic acid, and sodium ascorbate (L-Asc-Na) were purchased from Kanto Chemical (Tokyo, Japan). Sodium 2-(((butylthio)carbonothioyl)thio)propanoate (SBTPA),[1] galactose acrylamide,[2] and Neu5Ac acrylamide[2] were prepared according to previous papers. Commercial monomers including the radical inhibitor were purified by passing through an alumina column prior to use.

Proton nuclear resonance (^{1}H NMR) spectra were recorded on a JEOL-ECP400 spectrometer (JEOL, Tokyo, Japan) using D_2O as a solvent. Size exclusion chromatography (SEC) analysis was performed on a JASCO DG-980-50 degasser equipped with a JASCO PU-980 pump (JASCO Co., Tokyo, Japan), a Superdex 200 Increase 10/300 GL column (Cytiva, U.S.), a JASCO RI-2031 Plus RI detector. The analysis was performed at a flow rate of 0.5 mL/min by injecting 20 µL of a polymer solution (3 g/L) in 100 mM $NaNO_3$ aqueous solution (25 °C). All the samples were previously filtered through a 0.22 µm filter.

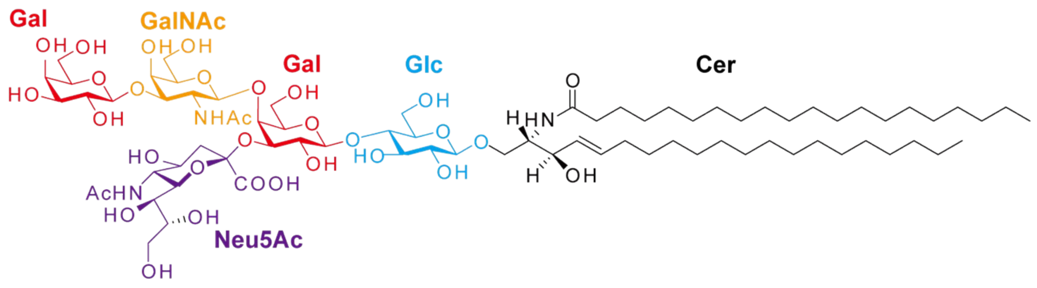
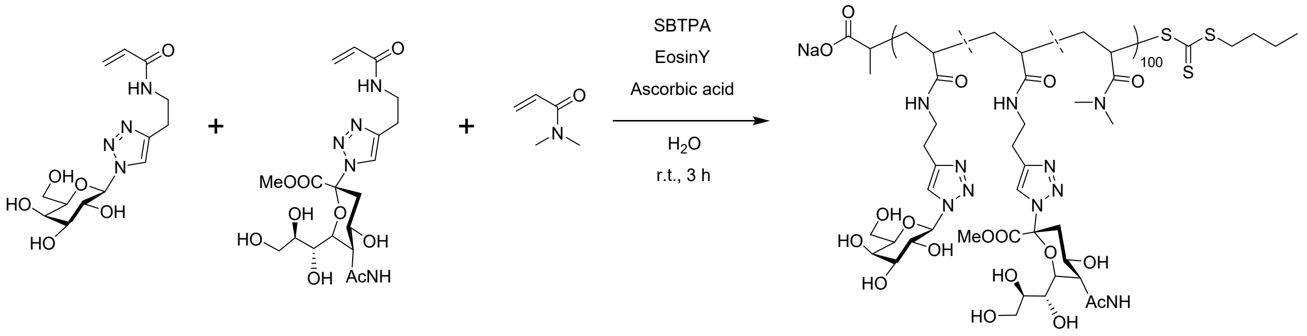
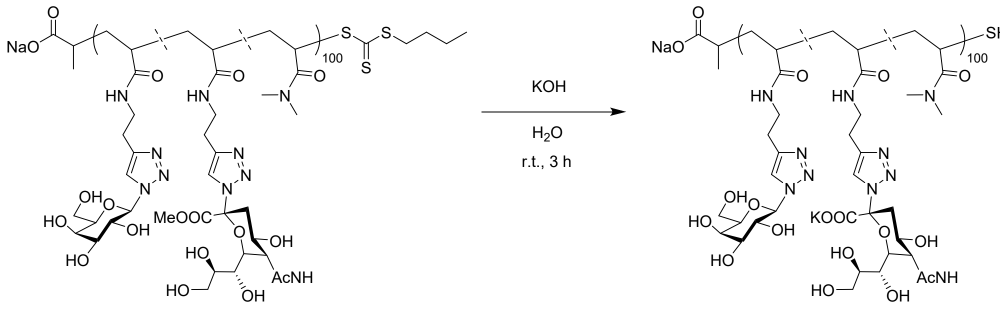


Figure S1. Molecular structure of GM1 ganglioside.

2. Synthetic procedure for glycopolymers by PET-RAFT polymerization



Three types of monomers (DMA, GalAAm, and Neu5AcAAm) were polymerized using PET-RAFT polymerization with reference to previous studies.[2] Setting the monomer concentration to 0.5 M, the monomers, RAFT agent (SBTPA), photooxidation-reduction catalyst (eosin Y), and reducing agent (ascorbic acid) were dissolved in Milli-Q water (200 μL) at the molar ratio of 100: 1: 0.01: 1. The mixture was put in a well of 96 well plate. The mixture was irradiated by LED lights (λ = 527 nm) at room temperature for 3 h. The conversion rate was determined by ^{1}H NMR, and relative molecular weight and dispersity (M_w/M_n) were calculated by SEC analysis.



Aqueous solution of potassium hydroxide (1 M, 30 μL) was added to each well (200 μL of polymer solution), and the mixture was incubated for 12 h at room temperature. The solution was neutralized with 1 M HCl solution. Then, the polymer solution was added to an ultrafiltration filter (MWCO: 3000), and purification was repeated 3 times in a centrifuge (14000 × g, 15 min). The filter tip was turned over and the sample was collected again in a centrifuge (1000 × g, 10 min) and then freeze-dried to obtain a solid glycopolymer sample.

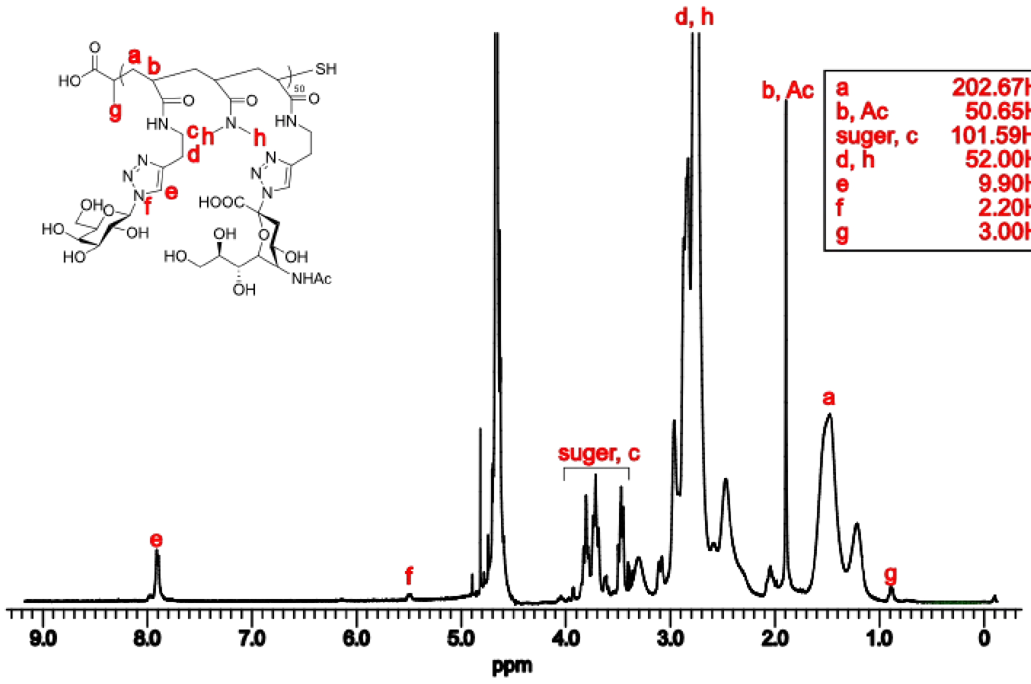


Figure S2. ^{1}H NMR spectrum of $\mathbf{G_4N_{16}D_{80}}$ (400 MHz, D_2O).

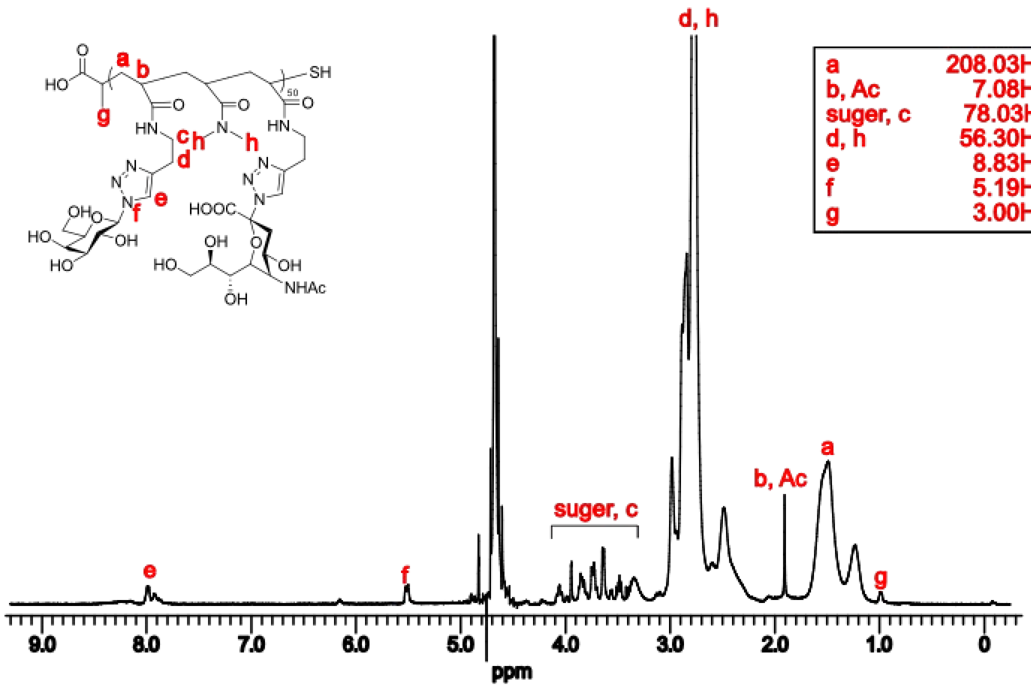


Figure S3. ^{1}H NMR spectrum of $\mathbf{G_{16}N_{64}D_{20}}$ (400 MHz, D_2O).

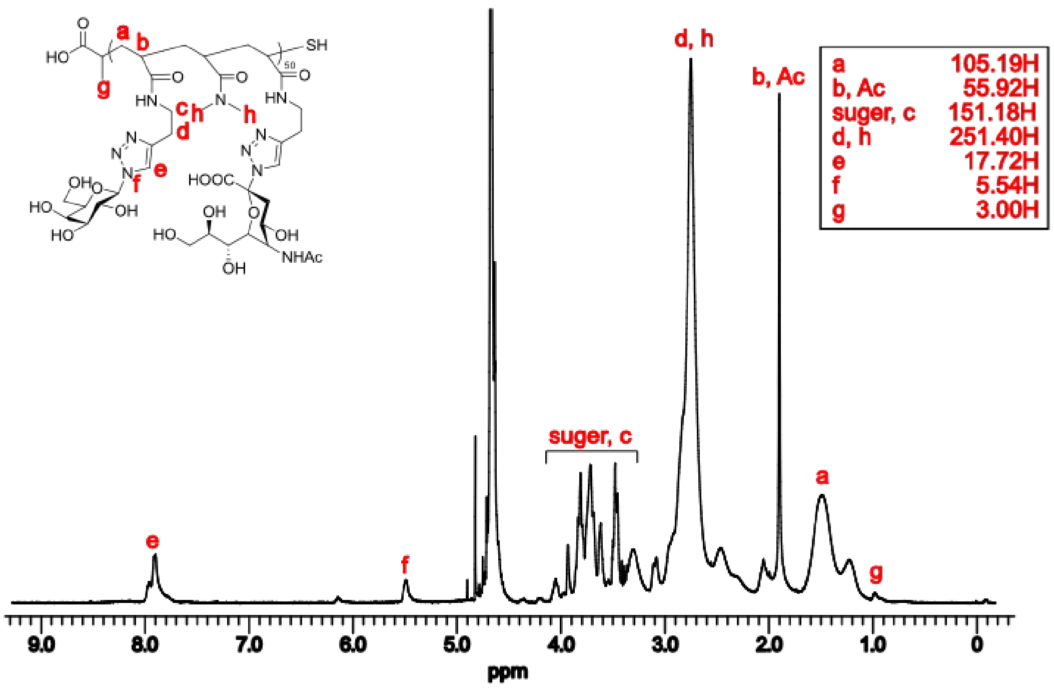


Figure S4. ^{1}H NMR spectrum of $\mathbf{G_{25}N_{25}D_{50}}$ (400 MHz, D_2O).

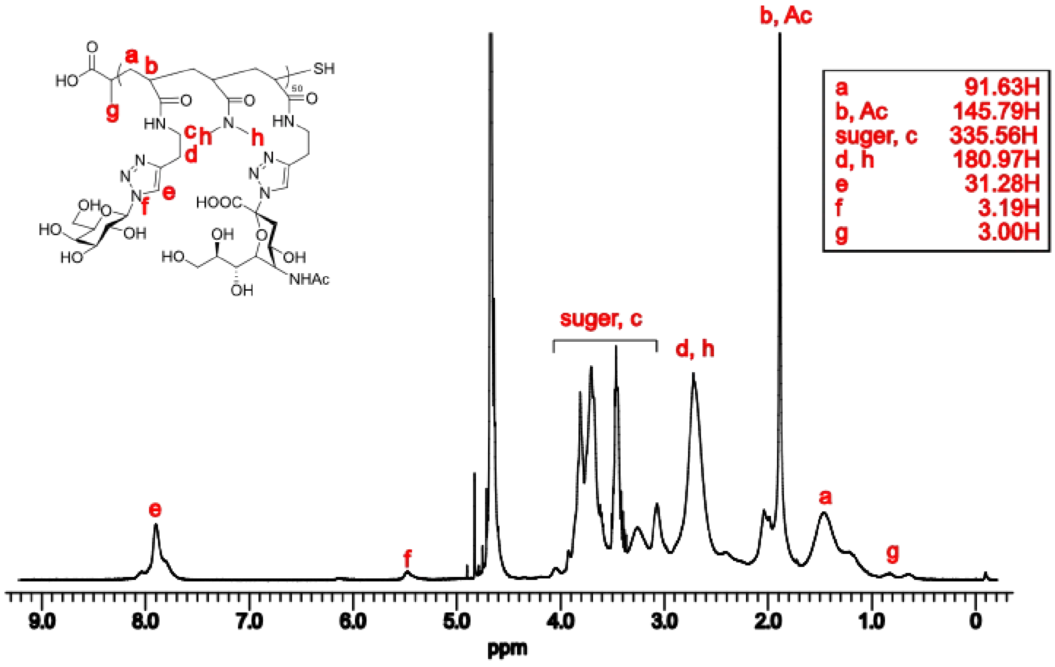


Figure S5. ^{1}H NMR spectrum of $\mathbf{G_{16}N_{4}D_{80}}$ (400 MHz, D_2O).

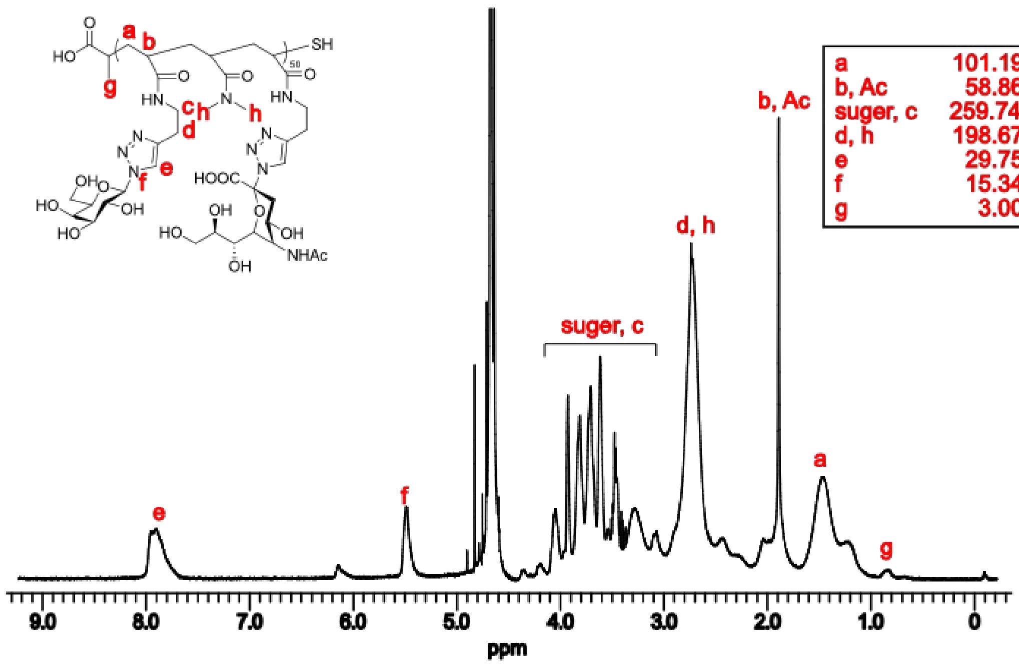


Figure S6. ^{1}H NMR spectrum of **$G_{64}N_{16}D_{20}$** (400 MHz, D_2O).

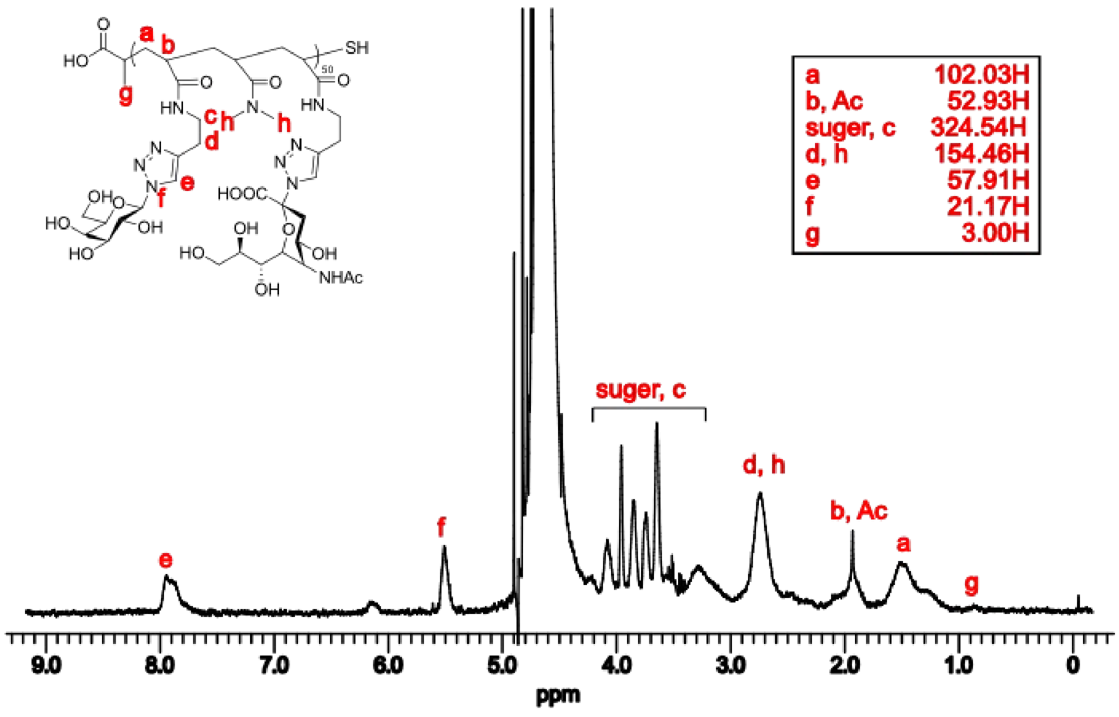


Figure S7. ^{1}H NMR spectrum of **$G_{90}N_{5}D_{5}$** (400 MHz, D_2O).

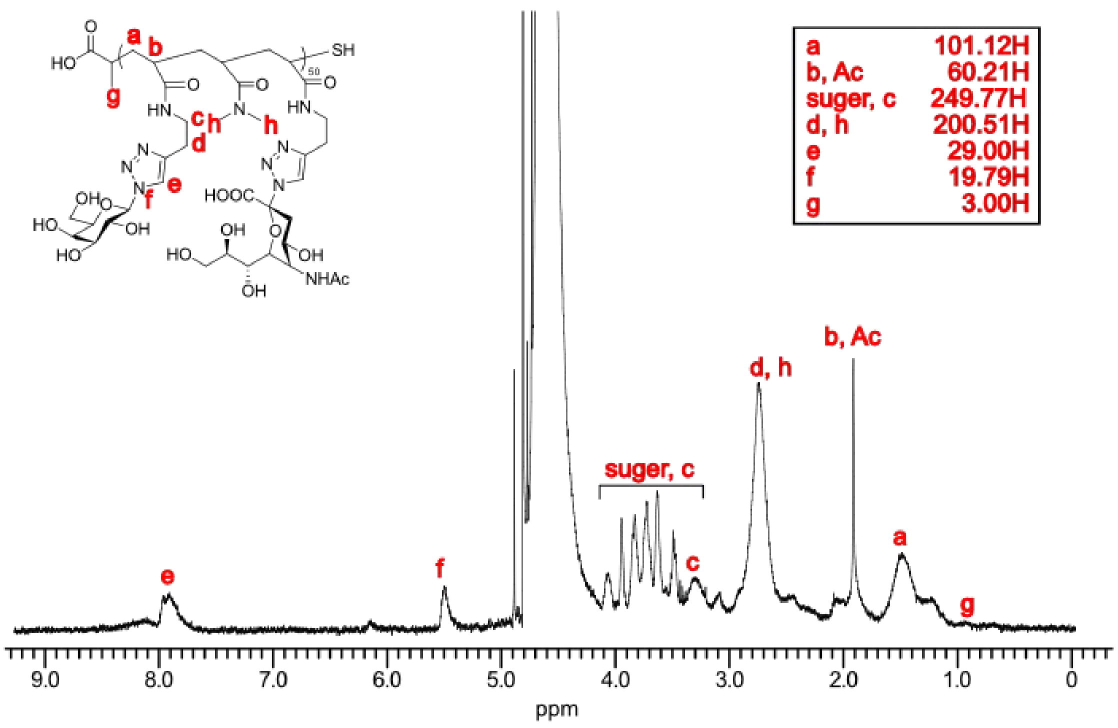


Figure S8. ${}^{1}H$ NMR spectrum of $\mathbf{G_{49}N_{21}D_{30}}$ (400 MHz, D_2O).

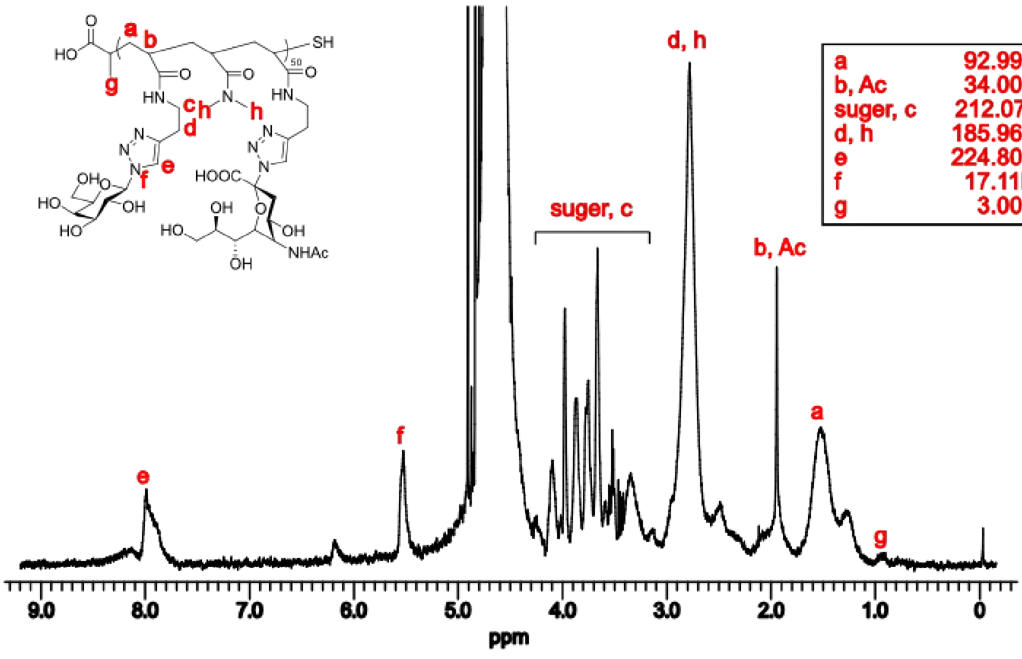


Figure S9. ${}^{1}H$ NMR spectrum of $\mathbf{G_{60}N_{10}D_{30}}$ (400 MHz, D_2O).

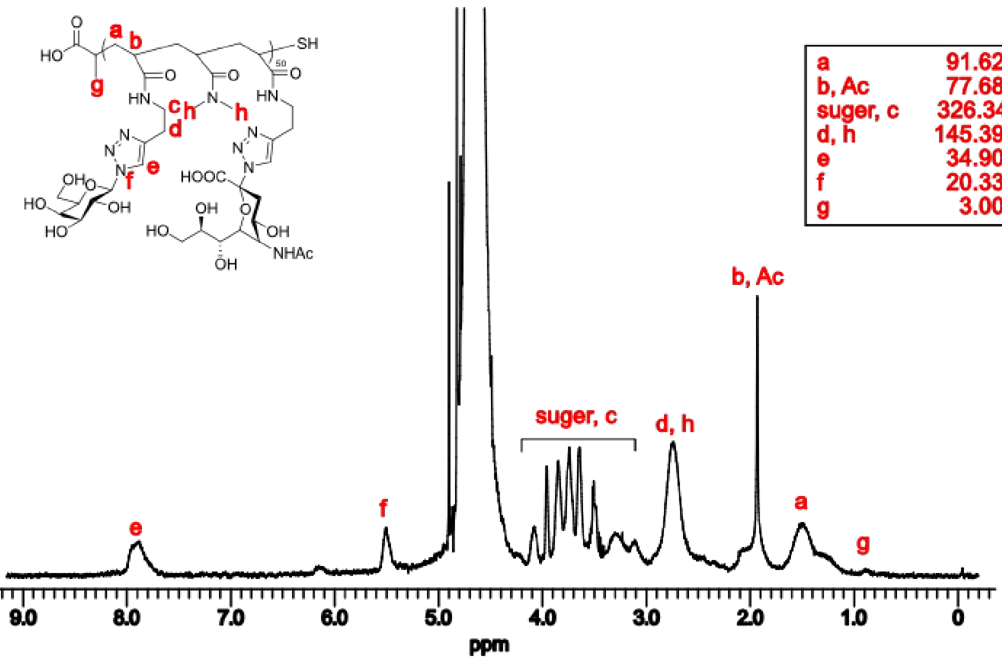


Figure S10. ^{1}H NMR spectrum of $\mathbf{G_{60}N_{25}D_{15}}$ (400 MHz, D_2O).

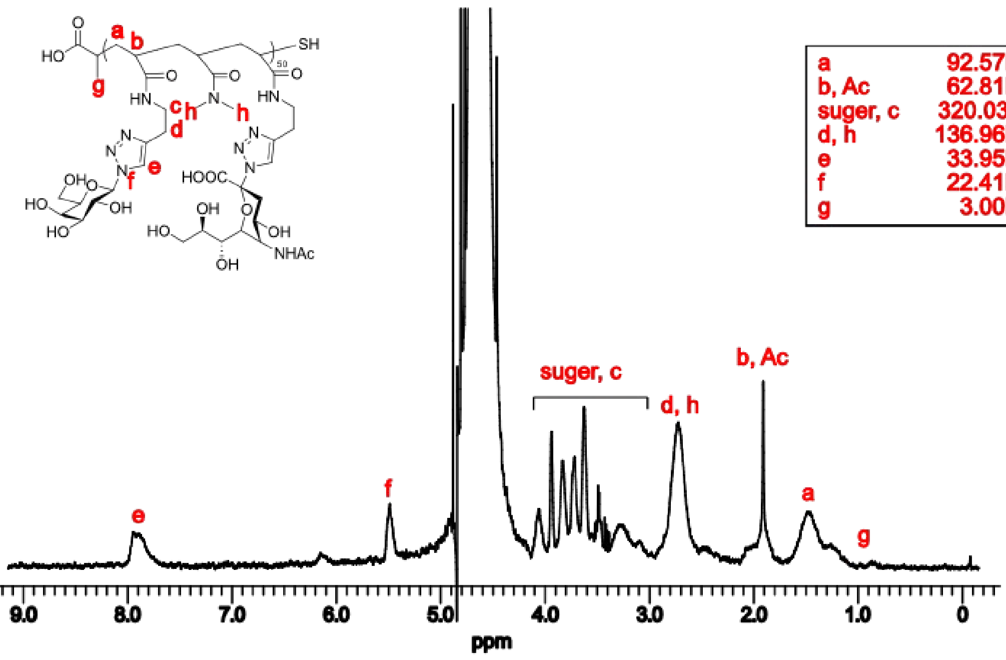


Figure S11. ^{1}H NMR spectrum of $\mathbf{G_{72}N_{13}D_{15}}$ (400 MHz, D_2O).

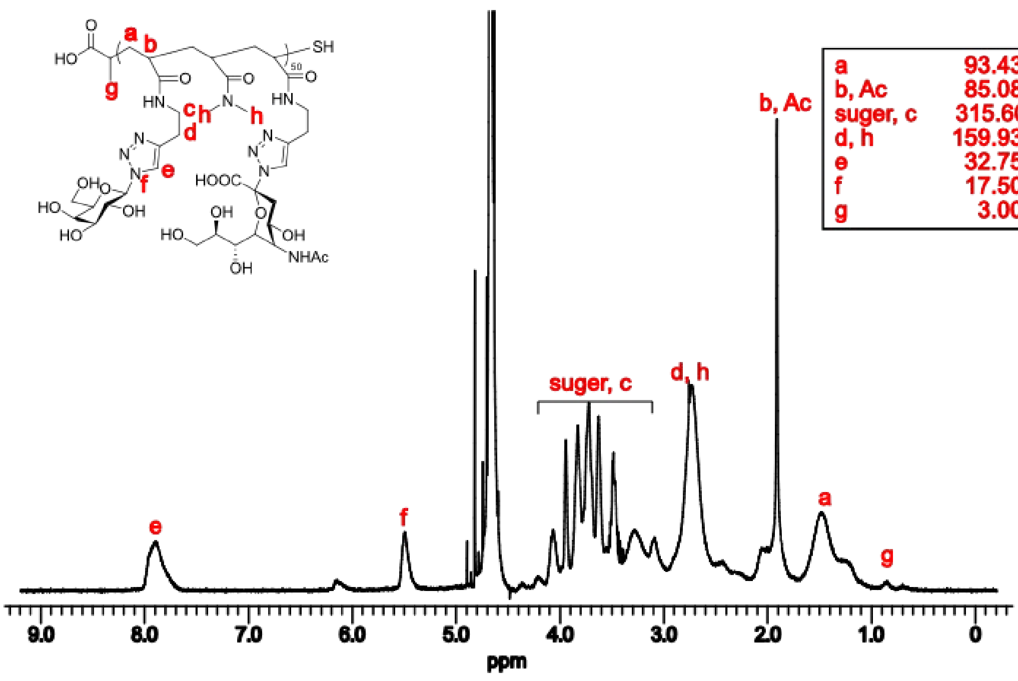


Figure S12. ^{1}H NMR spectrum of $\mathbf{G_{64}N_{27}D_9}$ (400 MHz, D_2O).

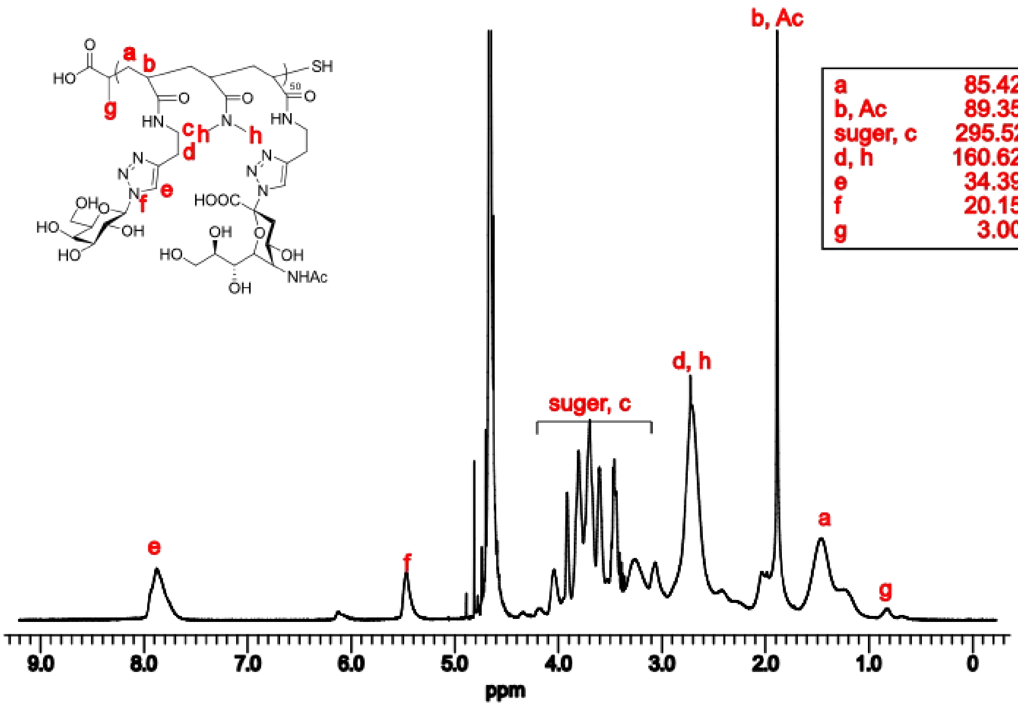


Figure S13. ^{1}H NMR spectrum of $\mathbf{G_{52}N_{31}D_{17}}$ (400 MHz, D_2O).

3. Enzyme-linked immunosorbent assay (ELISA)

Tris(hydroxymethyl)aminomethane (2.42 g, 0.02 mol), NaCl (3.23 g, 55 mmol), KCl (0.08 g, 1.1 mmol), and 200 µL of Tween 20 were dissolved in 400 mL of Milli-Q water to prepare an ELISA washing buffer. Each well of 96-well plate (Cert.Maxisorp Nunc-Immuno plate) was washed with 300 µL of the ELISA washing buffer (5 times). GM1 solution (0.1 µg/mL in PBS) was added to each well (100 µL) of a 96-well plate, and the plate was incubated overnight at 4°C. The GM1 solution in each well was discarded, and then each well was washed with 10 mM PBS five times. Subsequently, 300 µL of BSA solution (1 wt%) was added to each well, and the plate was left at room temperature (25°C) for 15 minutes. The glycopolymers were dissolved in a 1 wt% BSA solution to achieve a concentration of 0.5 mM and a total volume of 160 µL. An equal volume of CTB solution (10 ng/mL in 1 wt% BSA solution) was added to each glycopolymer solution to prepare a sample solution. Additionally, a positive control was prepared by mixing 200 µL of 0.1 µg/mL GM1 solution and CTB solution (10 ng/mL in 1 wt% BSA solution). A negative control was prepared by mixing 200 µL of 1 wt% BSA solution and CTB solution (10 ng/mL in 1 wt% BSA solution). The prepared sample solutions were incubated at 37°C for 2 hours. The BSA blocking solutions in the wells were discarded, and 100 µL of each sample solution, positive control solution, and negative control solution were added to three wells (n = 3), followed by incubation at 37°C for 30 minutes. The solutions were then discarded, and the wells were washed with washing buffer five times. Subsequently, 100 µL of primary antibody solution, diluted 1000 times in 1 wt% BSA solution, was added to each well, followed by incubation at 37°C for 1 hour. After discarding the primary antibody solution from the wells, the wells were washed with washing buffer five times. Finally, 100 µL of secondary antibody solution, diluted 5000 times in 1 wt% BSA solution, was added to each well, followed by incubation at 37°C for 1 hour. The TMB Microwell Peroxidase Substrate (2-Component System) was mixed in a 1:1 ratio (100 v/v %) and diluted with 10 mM PBS to prepare a 60 v/v % substrate solution. The secondary antibody in the wells was discarded, and the wells were washed five times with washing buffer. Then, 100 µL of the 60 v/v % substrate solution was added to each well, followed by incubation at 25°C for 20 minutes. The reaction was stopped by adding 100 µL of 1 M phosphoric acid solution to each well. Absorbance at 450 nm was measured using a spectral scanning multimode reader (Thermo).

Table S1. Results of ELISA at the 1st cycle.

Polymer	Absorbance (λ = 450 nm)	Normalized value*
$G_4N_{16}D_{80}$	1.74 ± 0.044	−0.029
$G_{16}N_{64}D_{20}$	1.61 ± 0.049	0.059
$G_{25}N_{25}D_{50}$	1.46 ± 0.009	0.15
$G_{16}N_4D_{80}$	1.27 ± 0.089	0.28
$G_{64}N_{16}D_{20}$	0.96 ± 0.029	0.48
Negative control (PBS)	1.70 ± 0.018	0
Positive control (GM1)	0.14 ± 0.004	1

*The normalized value was calculated following formula:

Normalized value = $(A_{\mathrm{Neg}} - A_{\mathrm{Polymer}}) / (A_{\mathrm{Neg}} - A_{\mathrm{Posi}})$

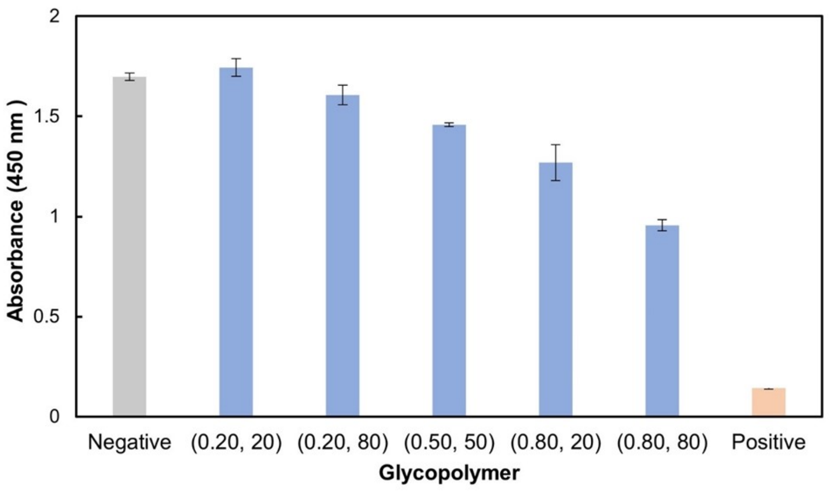


Figure S14. Absorbance at 450 nm obtained from ELISA (1st cycle).

Table S2. Results of ELISA at the 2nd cycle.

Polymer	Absorbance (λ = 450 nm)	Normalized value*
$G_{90}N_5D_5$	0.93 ± 0.032	0.33
Negative control (PBS)	1.33 ± 0.063	0
Positive control (GM1)	0.11 ± 0.003	1

*The normalized value was calculated following formula:

Normalized value = $(A_{\mathrm{Neg}}-A_{\mathrm{Polymer}})/(A_{\mathrm{Neg}}-A_{\mathrm{Posi}})$

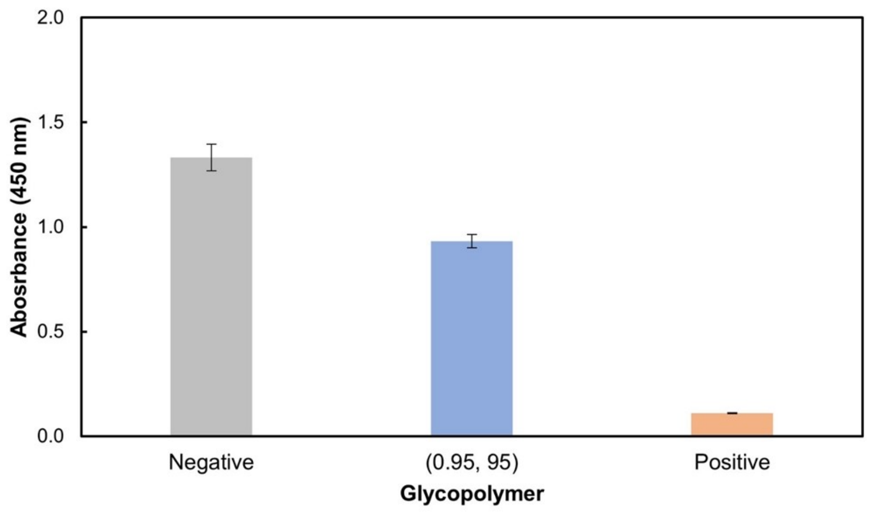


Figure S15. Absorbance at 450 nm obtained from ELISA (2nd cycle).

Table S3. Results of ELISA at the 3rd cycle.

Polymer	Absorbance (λ = 450 nm)	Normalized value*
$\mathbf{G_{49}N_{21}D_{30}}$	0.72 ± 0.021	0.50
$\mathbf{G_{60}N_{10}D_{30}}$	0.92 ± 0.096	0.34
$\mathbf{G_{60}N_{25}D_{15}}$	0.40 ± 0.027	0.77
$\mathbf{G_{72}N_{13}D_{15}}$	0.60 ± 0.036	0.60
Negative control (PBS)	1.33 ± 0.063	0
Positive control (GM1)	0.11 ± 0.003	1

*The normalized value was calculated following formula:

Normalized value = $(A_{\mathrm{Neg}} - A_{\mathrm{Polymer}}) / (A_{\mathrm{Neg}} - A_{\mathrm{Posi}})$

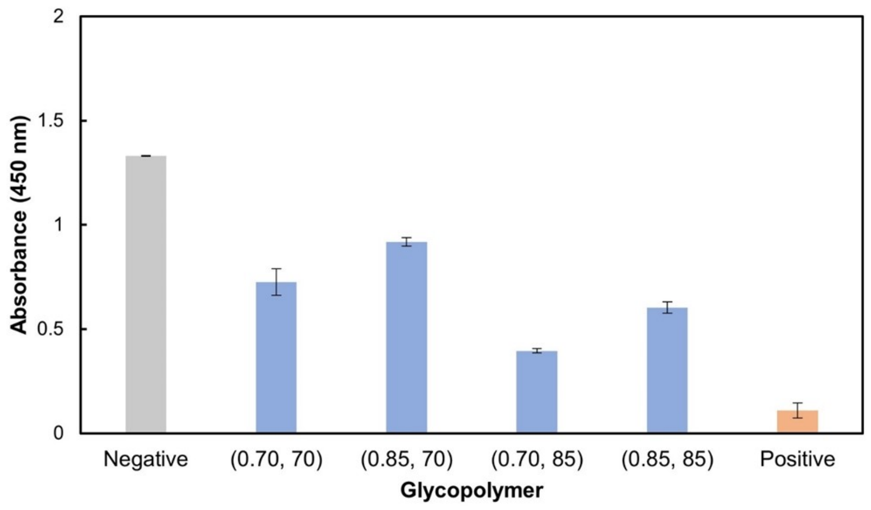


Figure S16. Absorbance at 450 nm obtained from ELISA (3rd cycle).

Table S4. Results of ELISA at the 4th cycle.

Polymer	Absorbance (λ = 450 nm)	Normalized value*
$G_{64}N_{27}D_9$	0.57 ± 0.027	0.65
Negative control (PBS)	1.41 ± 0.068	0
Positive control (GM1)	0.12 ± 0.005	1

*The normalized value was calculated following formula:

Normalized value = $(A_{\mathrm{Neg}}-A_{\mathrm{Polymer}})/(A_{\mathrm{Neg}}-A_{\mathrm{Posi}})$

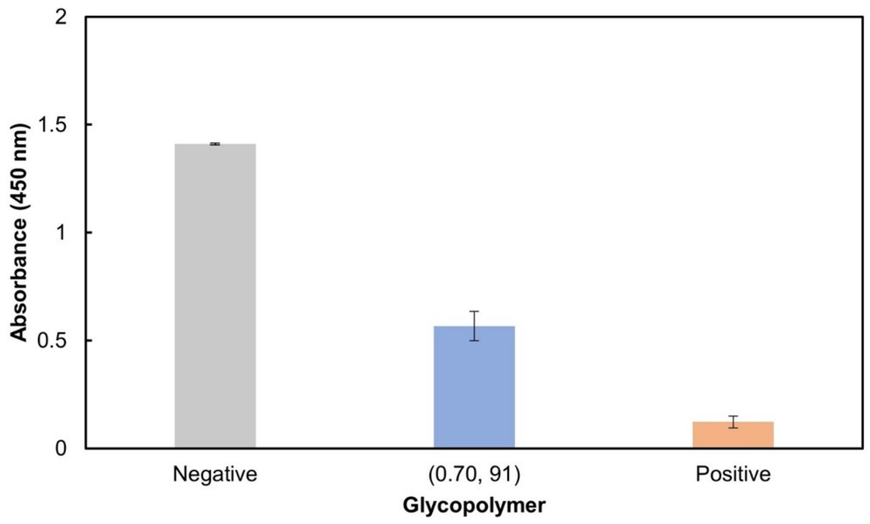


Figure S17. Absorbance at 450 nm obtained from ELISA (4th cycle).

Table S5. Results of ELISA at the 5th cycle.

Polymer	Absorbance (λ = 450 nm)	Normalized value*
$\mathbf{G_{52}N_{31}D_{17}}$	0.45 ± 0.017	0.71
Negative control (PBS)	1.28 ± 0.013	0
Positive control (GM1)	0.11 ± 0.004	1

*The normalized value was calculated following formula:

Normalized value = $(A_{\mathrm{Neg}}-A_{\mathrm{Polymer}})/\ (A_{\mathrm{Neg}}-A_{\mathrm{Posi}})$

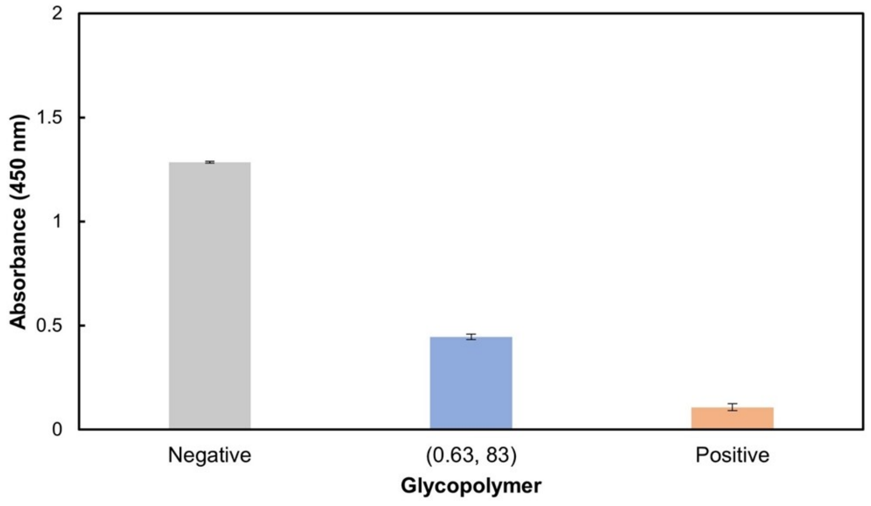


Figure S18. Absorbance at 450 nm obtained from ELISA (5th cycle).

4. Bayesian optimization by Gaussian process regression

Machine learning was employed using GPy model in Python to conduct Bayesian optimization of the glycopolymer structures.

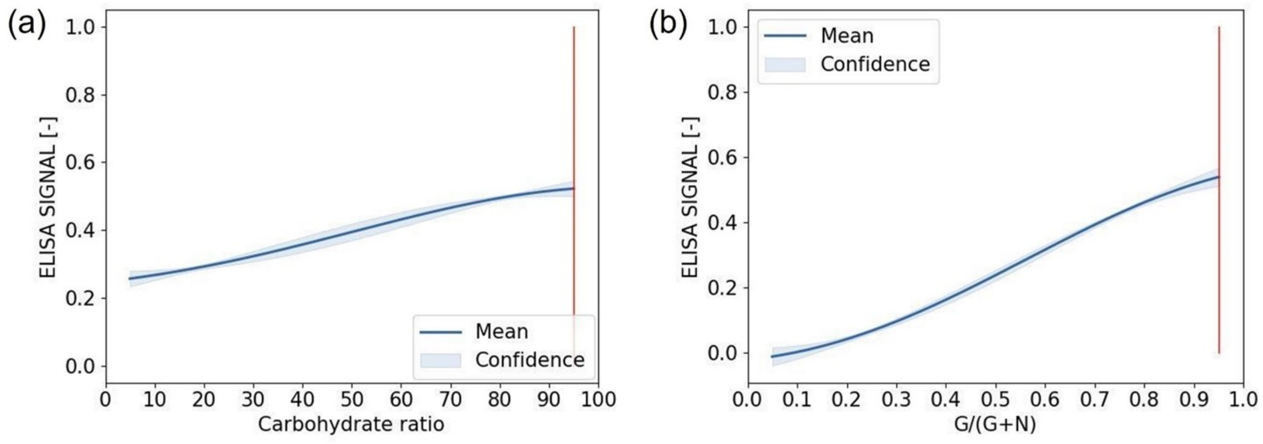


Figure S19. (a) Predicted yield for carbohydrate ratio in the yellow ring in Figure 2a. (b) Predicted yield for [G]/([G]+[N]) in the yellow ring in Figure 2a.

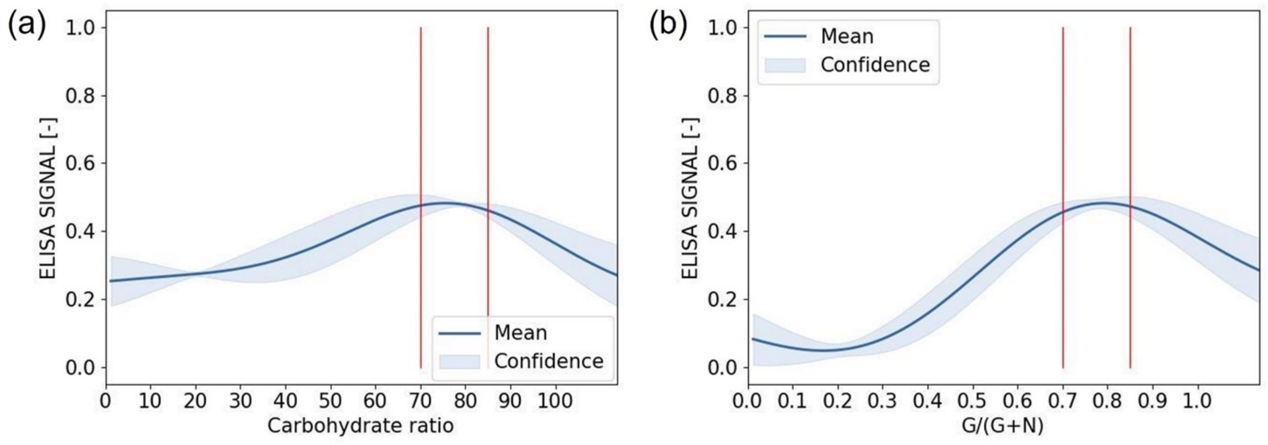


Figure S20. (a) Predicted yield for carbohydrate ratio in the yellow ring in Figure 2b. (b) Predicted yield for [G]/([G]+[N]) in the yellow ring in Figure 2b.

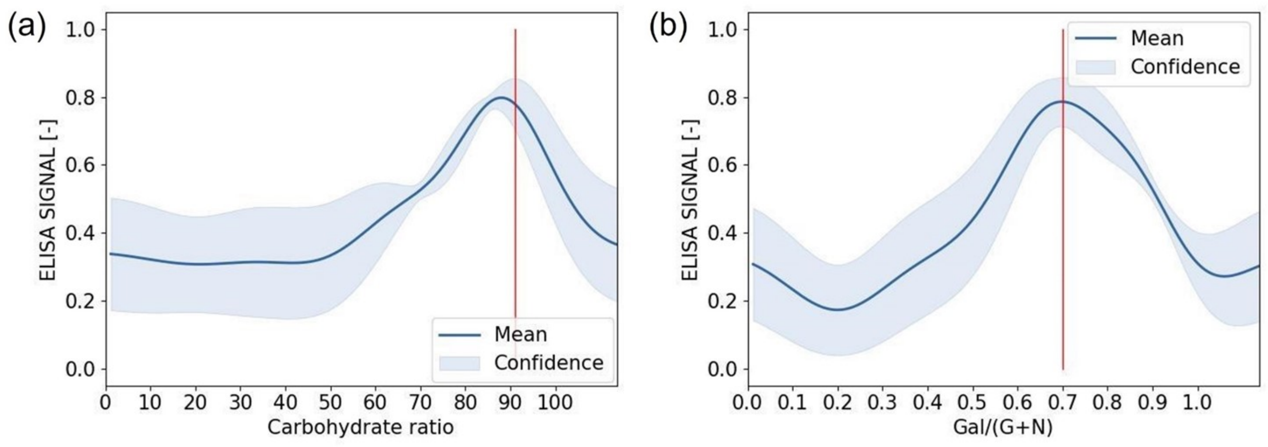


Figure S21. (a) Predicted yield for carbohydrate ratio in the yellow ring in Figure 2c. (b) Predicted yield for [G]/([G]+[N]) in the yellow ring in Figure 2c.

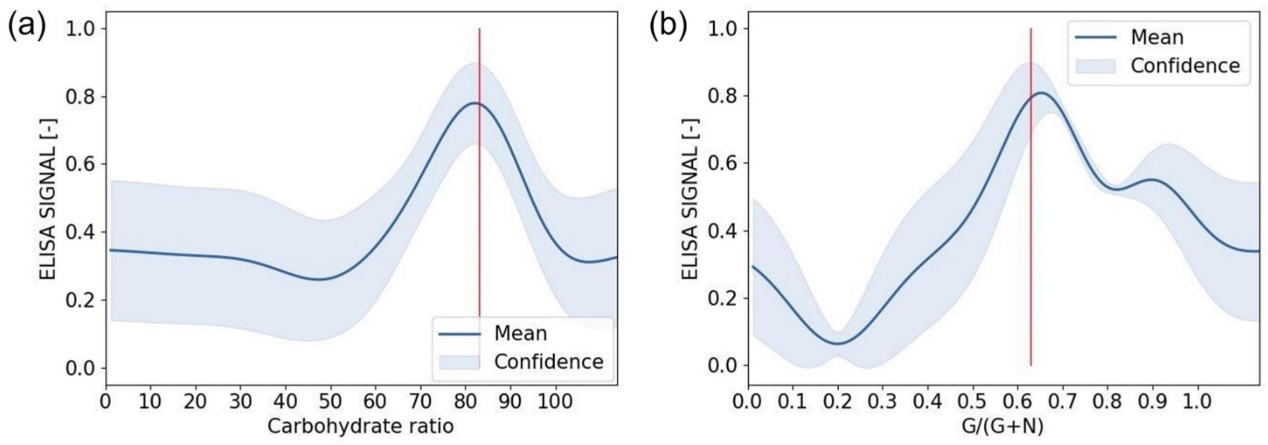


Figure S22. (a) Predicted yield for carbohydrate ratio in the yellow ring in Figure 2d. (b) Predicted yield for [G]/([G]+[N]) in the yellow ring in Figure 2d.

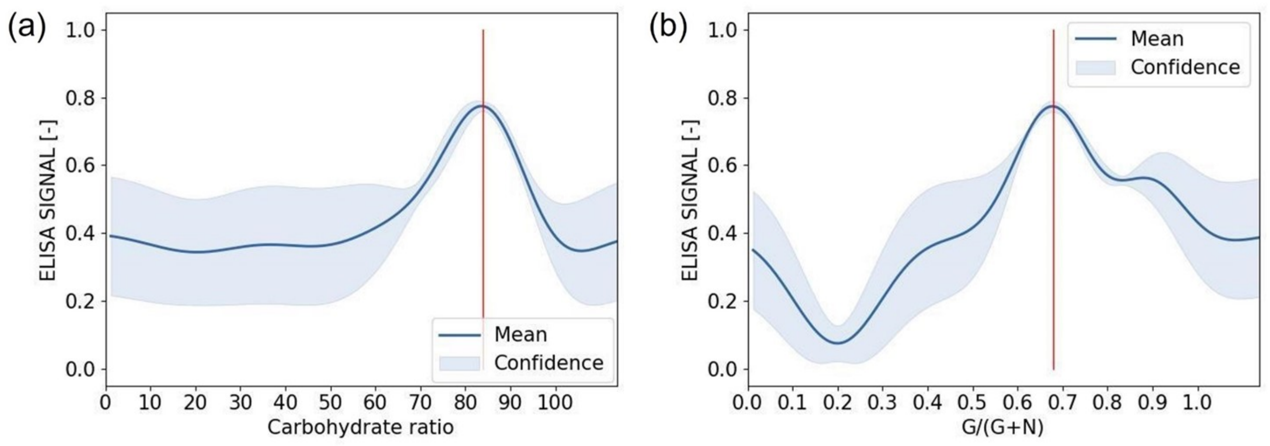


Figure S23. (a) Predicted yield for carbohydrate ratio in the yellow ring in Figure 2e. (b) Predicted yield for [G]/([G]+[N]) in the yellow ring in Figure 2e.

Code for Gaussian process regression (1st Cycle)

```
import GPy
import numpy as np
from matplotlib import pyplot as plt
import warnings
warnings.filterwarnings ('ignore')

#Gal/Neu5Ac, glyco rate
X= np.array([[0.200, 20],
              [0.200, 80],
              [0.500, 50],
              [0.800, 20],
              [0.800, 80]])

#yield
Y = np.array([-0.030, 0.059, 0.155, 0.276, 0.477])[:, np.newaxis]

#standardization
X_std = np.copy(X)
for i in range(X. shape[(1)]):
    X_std[:, i:i+1] = (X[:,i:i+1] - np.mean(X[:, i:i+1]))/np.sqrt(np.var(X[:, i:i+1]))
print(X_std)

#GPR and visualization
kernel = GPy.kern.RBF(input_dim=2)
model = GPy.models.GPRegression(X_std, Y, kernel=kernel, normalizer=True,
noise_var=0.001)
model.optimize(max_iters=3, messages=True)
model.plot()


x0_list_law = [0.00, 0.10, 0.20, 0.30, 0.40, 0.50, 0.60, 0.70, 0.80, 0.90, 1.00]
x1_list_law = [0, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
x0_list_std =[]
x1_list_std =[]
x0_list_std = (np.array(x0_list_law) - np.mean(X[:, 0:1])) / np.sqrt(np.var(X[:, 0:1]))
x1_list_std = (np.array(x1_list_law) - np.mean(X[:, 1:2])) / np.sqrt(np.var(X[:, 1:2]))
```

```
plt.xticks(x0_list_std, x0_list_law)
plt.yticks(x1_list_std, x1_list_law)
plt.xlabel("G/(G+N)", fontsize=15)
plt.ylabel("Carbohydrate ratio", fontsize=15)
plt.tick_params(labelsize=15)
plt.legend(bbox_to_anchor=(1,0.5), fontsize=15)


model.plot(fixed_inputs=[(1, 0.908)], plot_data=False, lower=25, upper=75)
x0_list_law = [0.00, 0.10, 0.20, 0.30, 0.40, 0.50, 0.60, 0.70, 0.80, 0.90, 1.00]
x0_list_std = []
x0_list_std = (np.array(x0_list_law) - np.mean(X[:, 0:1])) / np.sqrt(np.var(X[:, 0:1]))
x0_line = 0.95
x0_line_std = (x0_line - np.mean(X[:, 0:1])) / np.sqrt(np.var(X[:, 0:1]))

plt.xticks(x0_list_std, x0_list_law)
plt.xlabel("G/(G+N)", fontsize=15)
plt.ylabel("ELISA SIGNAL [-]", fontsize=15)
plt.tick_params(labelsize=15)
plt.legend(("Mean", "Confidence"), fontsize=15)
plt.legend(("Mean","Confidence",str(x0_line)+"%"),fontsize=15)
plt.plot([x0_line_std,x0_line_std],[0, 1.0],color='r',linewidth=1,linestyle='-',label=str(x0_line)+"%")

model.plot(fixed_inputs=[(0, 1.231)], plot_data=False, lower=25, upper=75)
x1_list_law =[0, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
x1_list_std =[]
x1_list_std = (np.array(x1_list_law) - np.mean(X[:, 1:2])) / np.sqrt(np.var(X[:, 1:2]))
plt.xticks(x1_list_std, x1_list_law)
x1_line = 95
x1_line_std =(x1_line - np.mean(X[:, 1:2])) / np.sqrt(np.var(X[:, 1:2]))

plt.xlabel("Carbohydrate ratio", fontsize=15)
plt.ylabel("ELISA SIGNAL [-]", fontsize=15)
plt.tick_params(labelsize=15)
plt.legend(("Mean", "Confidence"), loc="lower right", borderaxespad=0.2, fontsize=15)
plt.plot([x1_line_std,x1_line_std],[0, 1.0],color='r',linewidth=1,linestyle='-',label=str(x0_line)+"%")

plt.show()
```

Code for Gaussian process regression (2nd Cycle)

```
import GPy
import numpy as np
from matplotlib import pyplot as plt
import warnings
warnings.filterwarnings ('ignore')

#Gal/Neu5Ac, glyco rate
X= np.array([[0.200, 20],
                [0.200, 80],
                [0.500, 50],
                [0.800, 20],
                [0.800, 80],
                [0.950, 95]])

#yield
Y = np.array([-0.030, 0.059, 0.155, 0.276, 0.477, 0.327])[:, np.newaxis]

#standardization
X_std = np.copy(X)
for i in range(X. shape[(1)]):
    X_std[:, i:i+1] = (X[:,i:i+1] - np.mean(X[:, i:i+1]))/np.sqrt(np.var(X[:, i:i+1]))
print(X_std)

#GPR and visualization
kernel = GPy.kern.RBF(input_dim=2)
model = GPy.models.GPRegression(X_std, Y, kernel=kernel, normalizer=True,
noise_var=0.001)
model.optimize(max_iters=3, messages=True)
model.plot()


x0_list_law = [0.00, 0.10, 0.20, 0.30, 0.40, 0.50, 0.60, 0.70, 0.80, 0.90, 1.00]
x1_list_law = [0, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
x0_list_std =[]
x1_list_std =[]
x0_list_std = (np.array(x0_list_law) - np.mean(X[:, 0:1])) / np.sqrt(np.var(X[:, 0:1]))
```

```
x1_list_std = (np.array(x1_list_law) - np.mean(X[:, 1:2])) / np.sqrt(np.var(X[:, 1:2]))
plt.xticks(x0_list_std, x0_list_law)
plt.yticks(x1_list_std, x1_list_law)
plt.xlabel("G/(G+N)", fontsize=15)
plt.ylabel("Carbohydrate ratio", fontsize=15)
plt.tick_params(labelsize=15)
plt.legend(bbox_to_anchor=(1,0.5), fontsize=15)

model.plot(fixed_inputs=[(1, 0.605)], plot_data=False, lower=25, upper=75)
x0_list_law = [0.00, 0.10, 0.20, 0.30, 0.40, 0.50, 0.60, 0.70, 0.80, 0.90, 1.00]
x0_list_std = []
x0_list_std = (np.array(x0_list_law) - np.mean(X[:, 0:1])) / np.sqrt(np.var(X[:, 0:1]))
x0_line = 0.70, 0.85
x0_line_std = (x0_line - np.mean(X[:, 0:1])) / np.sqrt(np.var(X[:, 0:1]))

plt.xticks(x0_list_std, x0_list_law)
plt.xlabel("G/(G+N)", fontsize=15)
plt.ylabel("ELISA SIGNAL [-]", fontsize=15)
plt.tick_params(labelsize=15)
plt.legend(("Mean", "Confidence"), fontsize=15)
plt.legend(("Mean","Confidence",str(x0_line)+"%"),fontsize=15)
plt.plot([x0_line_std,x0_line_std],[0, 1.0],color='r',linewidth=1,linestyle='-',label=str(x0_line)+"%")

model.plot(fixed_inputs=[(0, 0.722)], plot_data=False, lower=25, upper=75)
x1_list_law =[0, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
x1_list_std =[]
x1_list_std = (np.array(x1_list_law) - np.mean(X[:, 1:2])) / np.sqrt(np.var(X[:, 1:2]))
plt.xticks(x1_list_std, x1_list_law)
x1_line = 70, 85
x1_line_std =(x1_line - np.mean(X[:, 1:2])) / np.sqrt(np.var(X[:, 1:2]))

plt.xlabel("Carbohydrate ratio", fontsize=15)
plt.ylabel("ELISA SIGNAL [-]", fontsize=15)
plt.tick_params(labelsize=15)
plt.legend(("Mean", "Confidence"), loc="lower right", borderaxespad=0.2, fontsize=15)
plt.plot([x1_line_std,x1_line_std],[0, 1.0],color='r',linewidth=1,linestyle='-',label=str(x0_line)+"%")

plt.show()
```

Code for Gaussian process regression (3rd Cycle)

```
import GPy
import numpy as np
from matplotlib import pyplot as plt
import warnings
warnings.filterwarnings ('ignore')

#Gal/Neu5Ac, glyco rate
X= np.array([[0.200, 20],
                    [0.200, 80],
                    [0.500, 50],
                    [0.800, 20],
                    [0.800, 80],
                    [0.950, 95],
                    [0.700, 70],
                    [0.700, 85],
                    [0.850, 70],
                    [0.850, 85]])

#yield
Y = np.array([-0.030, 0.059, 0.155, 0.276, 0.477, 0.327, 0.497, 0.766, 0.338, 0.596])[:, np.newaxis]

#standardization
X_std = np.copy(X)
for i in range(X. shape[(1)]):
    X_std[:, i:i+1] = (X[:,i:i+1] - np.mean(X[:, i:i+1]))/np.sqrt(np.var(X[:, i:i+1]))
print(X_std)

#GPR and visualization
kernel = GPy.kern.RBF(input_dim=2)
model = GPy.models.GPRegression(X_std, Y, kernel=kernel, normalizer=True,
noise_var=0.001)
model.optimize(max_iters=3, messages=True)
model.plot()

x0_list_law = [0.00, 0.10, 0.20, 0.30, 0.40, 0.50, 0.60, 0.70, 0.80, 0.90, 1.00]
x1_list_law = [0, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
```

```
x0_list_std =[]
x1_list_std =[]
x0_list_std = (np.array(x0_list_law) - np.mean(X[:, 0:1])) / np.sqrt(np.var(X[:, 0:1]))
x1_list_std = (np.array(x1_list_law) - np.mean(X[:, 1:2])) / np.sqrt(np.var(X[:, 1:2]))
plt.xticks(x0_list_std, x0_list_law)
plt.yticks(x1_list_std, x1_list_law)
plt.xlabel("G/(G+N)", fontsize=15)
plt.ylabel("Carbohydrate ratio", fontsize=15)
plt.tick_params(labelsize=15)
plt.legend(bbox_to_anchor=(1,0.5), fontsize=15)

model.plot(fixed_inputs=[(1, 0.997)], plot_data=False, lower=25, upper=75)
x0_list_law = [0.00, 0.10, 0.20, 0.30, 0.40, 0.50, 0.60, 0.70, 0.80, 0.90, 1.00]
x0_list_std = []
x0_list_std = (np.array(x0_list_law) - np.mean(X[:, 0:1])) / np.sqrt(np.var(X[:, 0:1]))
x0_line = 0.70
x0_line_std = (x0_line - np.mean(X[:, 0:1])) / np.sqrt(np.var(X[:, 0:1]))

plt.xticks(x0_list_std, x0_list_law)
plt.xlabel("Gal/(G+N)", fontsize=15)
plt.ylabel("ELISA SIGNAL [-]", fontsize=15)
plt.tick_params(labelsize=15)
plt.legend(("Mean", "Confidence"), fontsize=15)
plt.legend(("Mean","Confidence",str(x0_line)+"%"),fontsize=15)
plt.plot([x0_line_std,x0_line_std],[0, 1.0],color='r',linewidth=1,linestyle='-',label=str(x0_line)+"%")

model.plot(fixed_inputs=[(0, 0.097)], plot_data=False, lower=25, upper=75)
x1_list_law =[0, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
x1_list_std =[]
x1_list_std = (np.array(x1_list_law) - np.mean(X[:, 1:2])) / np.sqrt(np.var(X[:, 1:2]))
plt.xticks(x1_list_std, x1_list_law)
x1_line = 91
x1_line_std =(x1_line - np.mean(X[:, 1:2])) / np.sqrt(np.var(X[:, 1:2]))

plt.xlabel("Carbohydrate ratio", fontsize=15)
plt.ylabel("ELISA SIGNAL [-]", fontsize=15)
plt.tick_params(labelsize=15)
plt.legend(("Mean", "Confidence"), loc="lower right", borderaxespad=0.2, fontsize=15)
plt.plot([x1_line_std,x1_line_std],[0, 1.0],color='r',linewidth=1,linestyle='-',label=str(x0_line)+"%")

plt.show()
```

Code for Gaussian process regression (4th Cycle)

```
import GPy
import numpy as np
from matplotlib import pyplot as plt
import warnings
warnings.filterwarnings ('ignore')

#Gal/Neu5Ac, glyco rate
X= np.array([[0.200, 20],
                [0.200, 80],
                [0.500, 50],
                [0.800, 20],
                [0.800, 80],
                [0.950, 95],
                [0.700, 70],
                [0.700, 85],
                [0.850, 70],
                [0.850, 85],
                [0.700, 91]])

#yield
Y = np.array([-0.030, 0.059, 0.155, 0.276, 0.477, 0.327, 0.497, 0.766, 0.338, 0.596, 0.655])[:,
np.newaxis]

#standardization
X_std = np.copy(X)
for i in range(X. shape[(1)]):
    X_std[:, i:i+1] = (X[:,i:i+1] - np.mean(X[:, i:i+1]))/np.sqrt(np.var(X[:, i:i+1]))
print(X_std)

#GPR and visualization
kernel = GPy.kern.RBF(input_dim=2)
model = GPy.models.GPRegression(X_std, Y, kernel=kernel, normalizer=True,
noise_var=0.001)
model.optimize(max_iters=3, messages=True)
model.plot()
```

```
x0_list_law = [0.00, 0.10, 0.20, 0.30, 0.40, 0.50, 0.60, 0.70, 0.80, 0.90, 1.00]
x1_list_law = [0, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
x0_list_std =[]
x1_list_std =[]
x0_list_std = (np.array(x0_list_law) - np.mean(X[:, 0:1])) / np.sqrt(np.var(X[:, 0:1]))
x1_list_std = (np.array(x1_list_law) - np.mean(X[:, 1:2])) / np.sqrt(np.var(X[:, 1:2]))
plt.xticks(x0_list_std, x0_list_law)
plt.yticks(x1_list_std, x1_list_law)
plt.xlabel("G/(G+N)", fontsize=15)
plt.ylabel("Carbohydrate ratio", fontsize=15)
plt.tick_params(labelsize=15)
plt.legend(bbox_to_anchor=(1,0.5), fontsize=15)

model.plot(fixed_inputs=[(1, 0.551)], plot_data=False, lower=25, upper=75)
x0_list_law = [0.00, 0.10, 0.20, 0.30, 0.40, 0.50, 0.60, 0.70, 0.80, 0.90, 1.00]
x0_list_std = []
x0_list_std = (np.array(x0_list_law) - np.mean(X[:, 0:1])) / np.sqrt(np.var(X[:, 0:1]))
x0_line = 0.63
x0_line_std = (x0_line - np.mean(X[:, 0:1])) / np.sqrt(np.var(X[:, 0:1]))

plt.xticks(x0_list_std, x0_list_law)
plt.xlabel("G/(G+N)", fontsize=15)
plt.ylabel("ELISA SIGNAL [-]", fontsize=15)
plt.tick_params(labelsize=15)
plt.legend(("Mean", "Confidence"), fontsize=15)
plt.legend(("Mean","Confidence",str(x0_line)+"%"),fontsize=15)
plt.plot([x0_line_std,x0_line_std],[0,1.0],color='r',linewidth=1,linestyle='-',label=str(x0_line)+"%")

model.plot(fixed_inputs=[(0, -0.161)], plot_data=False, lower=25, upper=75)
x1_list_law =[0, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
x1_list_std =[]
x1_list_std = (np.array(x1_list_law) - np.mean(X[:, 1:2])) / np.sqrt(np.var(X[:, 1:2]))
plt.xticks(x1_list_std, x1_list_law)
x1_line = 83
x1_line_std =(x1_line - np.mean(X[:, 1:2])) / np.sqrt(np.var(X[:, 1:2]))

plt.xlabel("Carbohydrate ratio", fontsize=15)
plt.ylabel("ELISA SIGNAL [-]", fontsize=15)
plt.tick_params(labelsize=15)
plt.legend(("Mean", "Confidence"), loc="lower right", borderaxespad=0.2, fontsize=15)
plt.plot([x1_line_std,x1_line_std],[0, 1.0],color='r',linewidth=1,linestyle='-',label=str(x0_line)+"%")

plt.show()
```

Code for Gaussian process regression (5th Cycle)

```
import GPy
import numpy as np
from matplotlib import pyplot as plt
import warnings
warnings.filterwarnings ('ignore')

#Gal/Neu5Ac, glyco rate
X= np.array([[0.200, 20],
             [0.200, 80],
             [0.500, 50],
             [0.800, 20],
             [0.800, 80],
             [0.950, 95],
             [0.700, 70],
             [0.700, 85],
             [0.850, 70],
             [0.850, 85],
             [0.700, 91],
             [0.630, 83]])

#yield
Y = np.array([-0.030, 0.059, 0.155, 0.276, 0.477, 0.327, 0.497, 0.766, 0.338, 0.596, 0.655, 0.713])[:,
np.newaxis]

#standardization
X_std = np.copy(X)
for i in range(X. shape[(1)]):
    X_std[:, i:i+1] = (X[:,i:i+1] - np.mean(X[:, i:i+1]))/np.sqrt(np.var(X[:, i:i+1]))
print(X_std)

#GPR and visualization
kernel = GPy.kern.RBF(input_dim=2)
model = GPy.models.GPRegression(X_std, Y, kernel=kernel, normalizer=True,
noise_var=0.001)
model.optimize(max_iters=3, messages=True)
model.plot()
```

```
x0_list_law = [0.00, 0.10, 0.20, 0.30, 0.40, 0.50, 0.60, 0.70, 0.80, 0.90, 1.00]
x1_list_law = [0, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
x0_list_std =[]
x1_list_std =[]
x0_list_std = (np.array(x0_list_law) - np.mean(X[:, 0:1])) / np.sqrt(np.var(X[:, 0:1]))
x1_list_std = (np.array(x1_list_law) - np.mean(X[:, 1:2])) / np.sqrt(np.var(X[:, 1:2]))
plt.xticks(x0_list_std, x0_list_law)
plt.yticks(x1_list_std, x1_list_law)
plt.xlabel("G/(G+N)", fontsize=15)
plt.ylabel("Carbohydrate ratio", fontsize=15)
plt.tick_params(labelsize=15)
plt.legend(bbox_to_anchor=(1,0.5), fontsize=15)

model.plot(fixed_inputs=[(1, 0.570)], plot_data=False, lower=25, upper=75)
x0_list_law = [0.00, 0.10, 0.20, 0.30, 0.40, 0.50, 0.60, 0.70, 0.80, 0.90, 1.00]
x0_list_std = []
x0_list_std = (np.array(x0_list_law) - np.mean(X[:, 0:1])) / np.sqrt(np.var(X[:, 0:1]))
x0_line = 0.68
x0_line_std = (x0_line - np.mean(X[:, 0:1])) / np.sqrt(np.var(X[:, 0:1]))

plt.xticks(x0_list_std, x0_list_law)
plt.xlabel("G/(G+N)", fontsize=15)
plt.ylabel("ELISA SIGNAL [-]", fontsize=15)
plt.tick_params(labelsize=15)
plt.legend(("Mean", "Confidence"), fontsize=15)
plt.legend(("Mean","Confidence",str(x0_line)+"%"),fontsize=15)
plt.plot([x0_line_std,x0_line_std],[0,1.0],color='r',linewidth=1,linestyle='-',label=str(x0_line)+"%")

model.plot(fixed_inputs=[(0, 0.090)], plot_data=False, lower=25, upper=75)
x1_list_law =[0, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100]
x1_list_std =[]
x1_list_std = (np.array(x1_list_law) - np.mean(X[:, 1:2])) / np.sqrt(np.var(X[:, 1:2]))
plt.xticks(x1_list_std, x1_list_law)
x1_line = 84
x1_line_std =(x1_line - np.mean(X[:, 1:2])) / np.sqrt(np.var(X[:, 1:2]))

plt.xlabel("Carbohydrate ratio", fontsize=15)
plt.ylabel("ELISA SIGNAL [-]", fontsize=15)
plt.tick_params(labelsize=15)
plt.legend(("Mean", "Confidence"), loc="lower right", borderaxespad=0.2, fontsize=15)
plt.plot([x1_line_std,x1_line_std],[0, 1.0],color='r',linewidth=1,linestyle='-',label=str(x0_line)+"%")

plt.show()
```

Reference:

(1) M. Nagao, M. Kichize, Y. Hoshino, and Y. Miura, *Biomacromolecules* **2021**, *22*, 3119–3127.

(2) M. Nagao, T. Uemura, T. Horiuchi, Y. Hoshino, and Y. Miura, *Chem. Commun.* **2021**, *57*, 10871–10874.