SUPPORTING INFORMATION

Table of Contents

Results and Discussion

S1. Comparison of cycloaddition kinetics between glycals

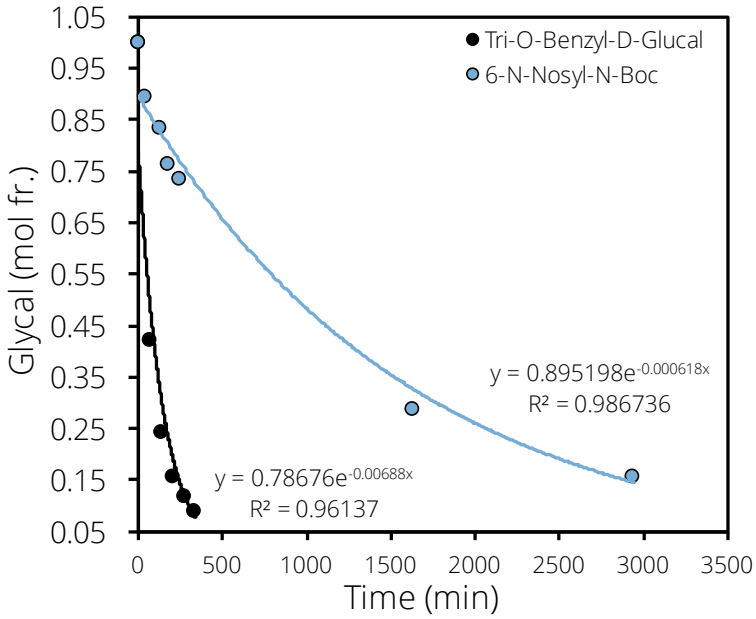


Figure S1. Consumption of glycal as monitored by NMR for reactions between trichloroacetyl isocyanate (TCAI) and tri-*O*-benzyl-D-glucal (black line) and TCAI and 3,4-di-*O*-benzyl-6-*N*-nosyl-*N*-boc-6-deoxy-D-glucal (blue line). The reaction points are fit to an exponential decay (first order kinetics) and the reaction rate is represented by the decay constant (-0.00688 for tri-*O*-benzyl-D-glucal, -0.000618 for 3,4-di-*O*-benzyl-6-*N*-nosyl-*N*-boc-6-deoxy-D-glucal). The rate of consumption of tri-*O*-benzyl-D-glucal is an order of magnitude higher than that of 3,4-di-*O*-benzyl-6-*N*-nosyl-*N*-boc-6-deoxy-D-glucal.

S2. ^{1}H-NMR spectra of P1''-P3''

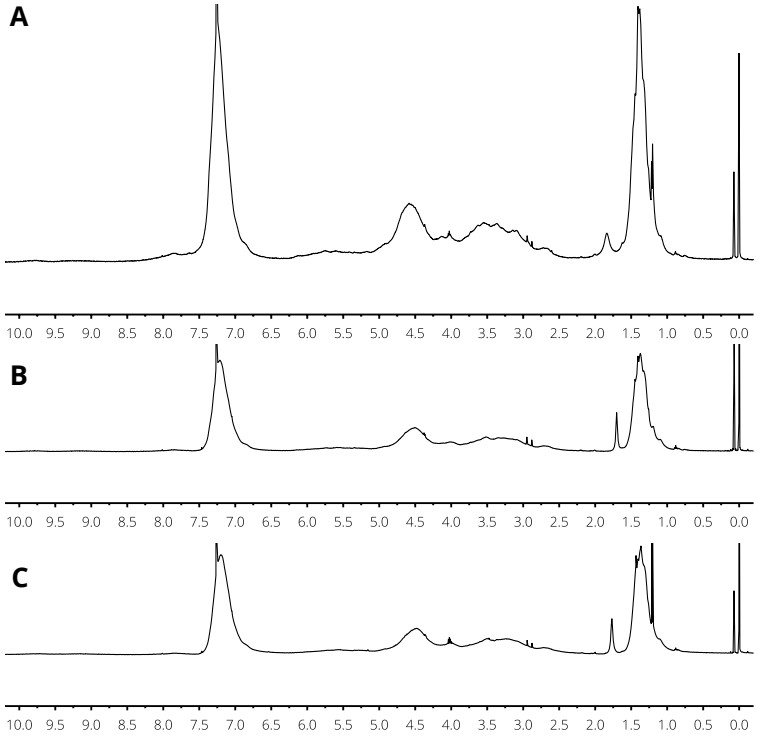


Figure S2. ^{1}H-NMR spectra of **P1''-P3''**. (A): **P1''**; (B): **P2''**; (C): **P3''**.

S3. ^{13}C-NMR spectra of P1''

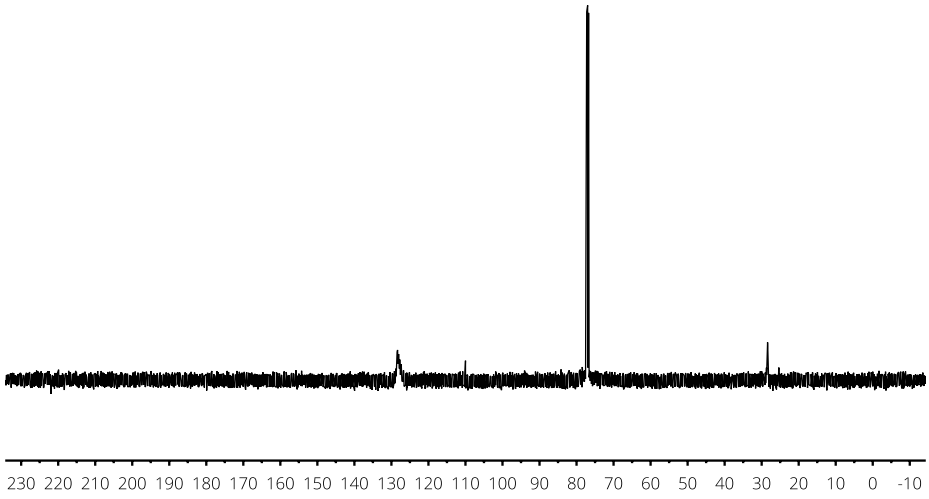


Figure S3. ^{13}C-NMR spectra of **P1''**

S4. ^{1}H-NMR spectra of P1-P3

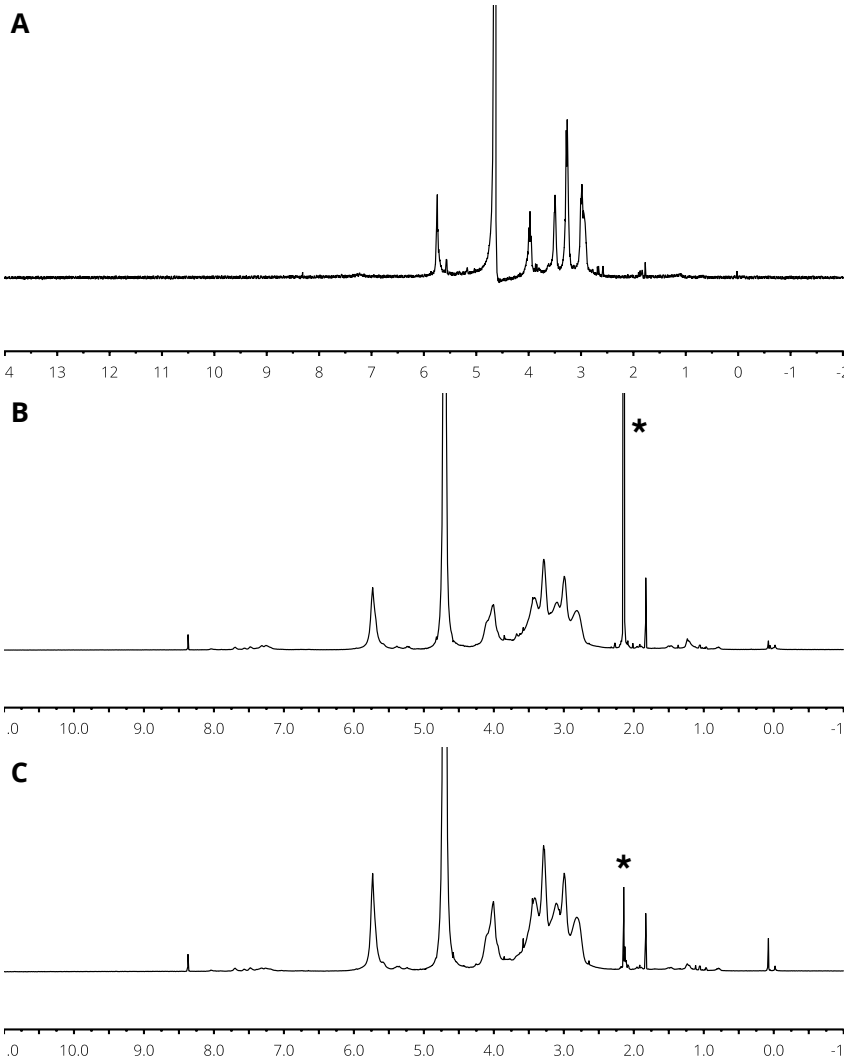


Figure S4. ^{1}H-NMR spectra of **P1**-**P3**. (A): **P1**; (B): **P2**; (C): **P3**. * - residual acetone.

S5. ^{13}C-NMR spectra of P1-P3

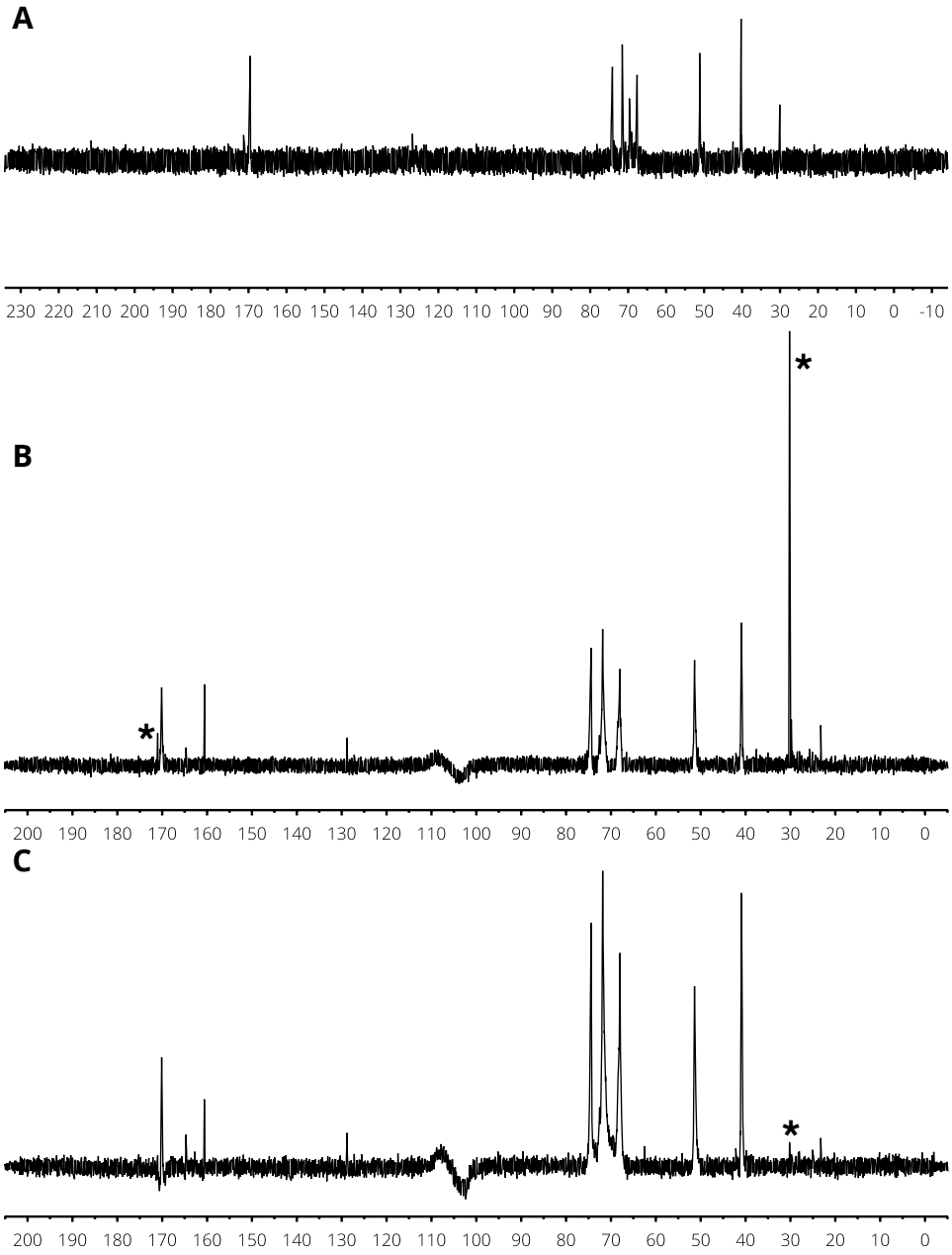


Figure S5. ^{13}C-NMR spectra of **P1-P3**. (A): **P1**; (B): **P2**; (C): **P3**. * - residual acetone.

S6. $^{1}H^{13}C$-HSQC spectra of P2 and P3

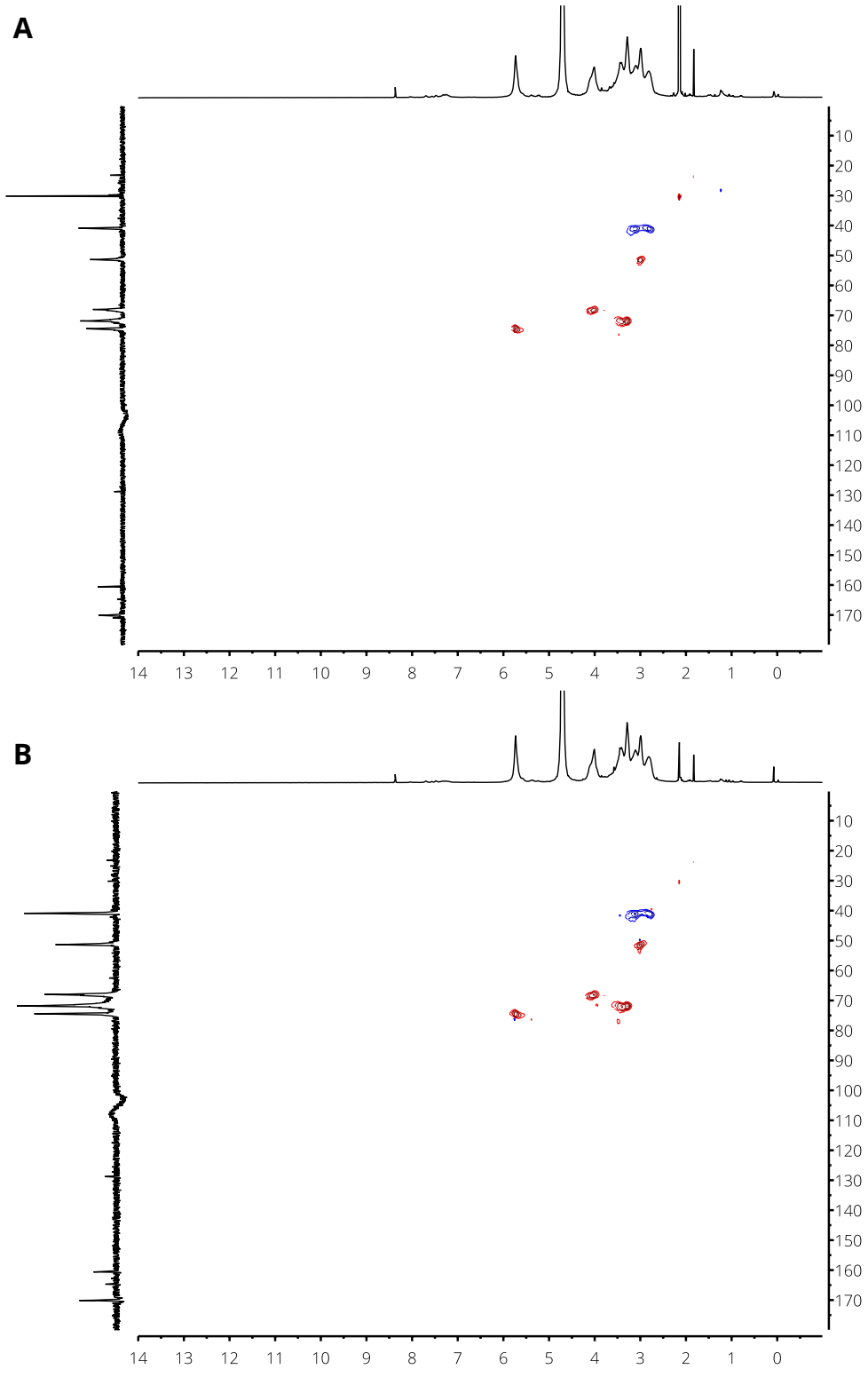


Figure S6. $^{1}H^{13}C$-HSQC spectra of **P2** and **P3**. (A): **P2**; (B): **P3**.

S7. IR spectra of P1''-P3''

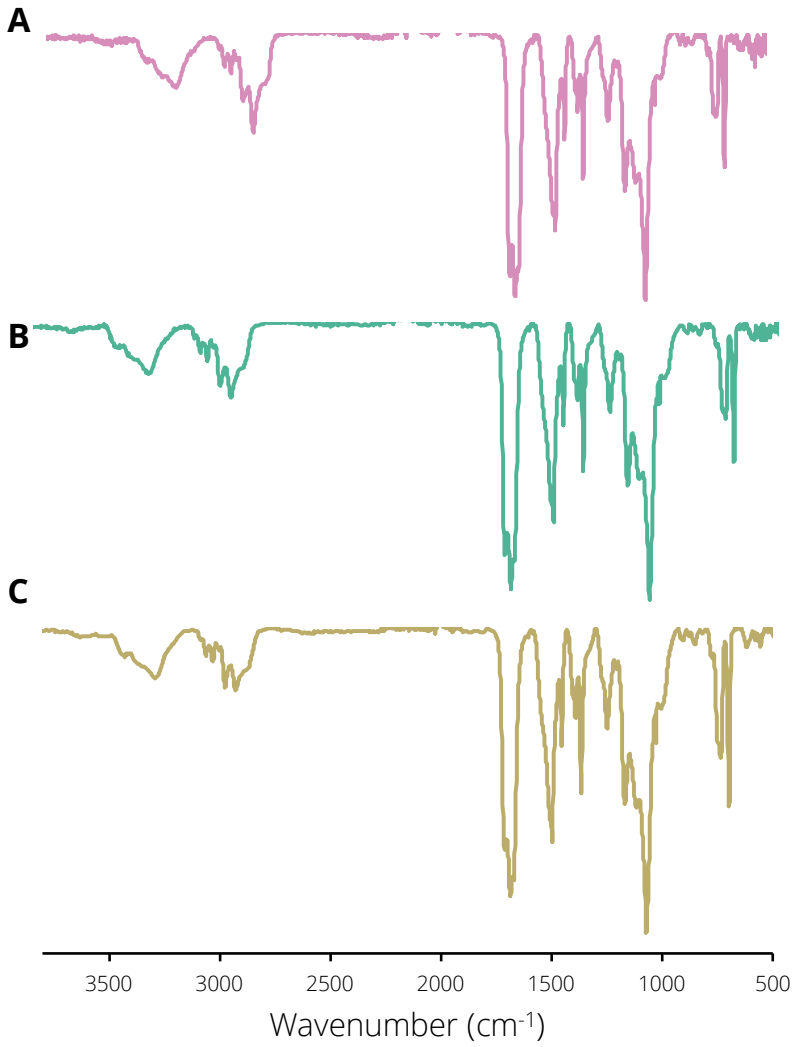


Figure S7. IR spectra of: (A) **P1''**; (B) **P2''**; (C) **P3''**

S8. Deprotected IR spectra

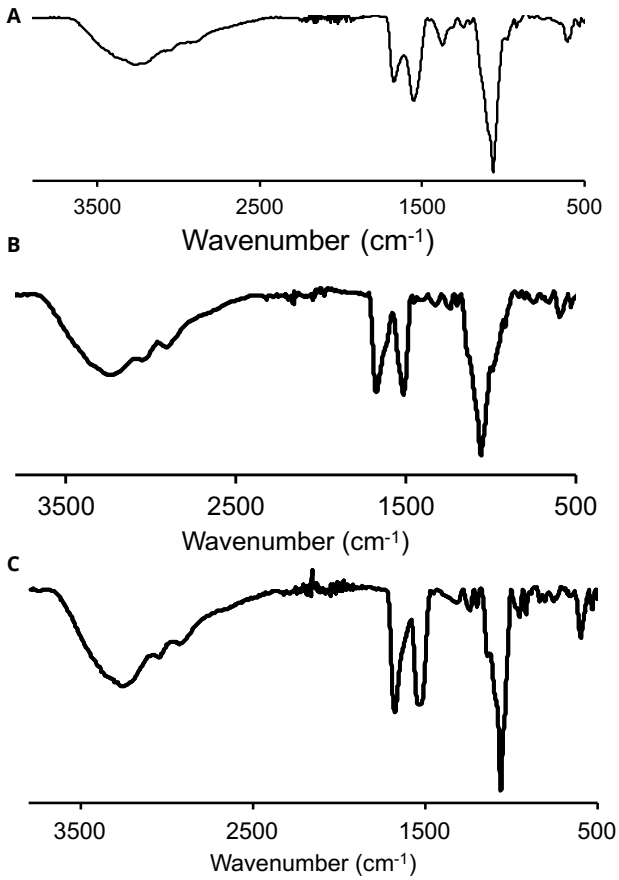


Figure S8. IR spectra of **P1-P3**. (A) **P1**; (B) **P2**; (C) **P3**.

S9. GPC Traces of Protected and Deprotected Polymers (P1''-P3'' and P1-P3)

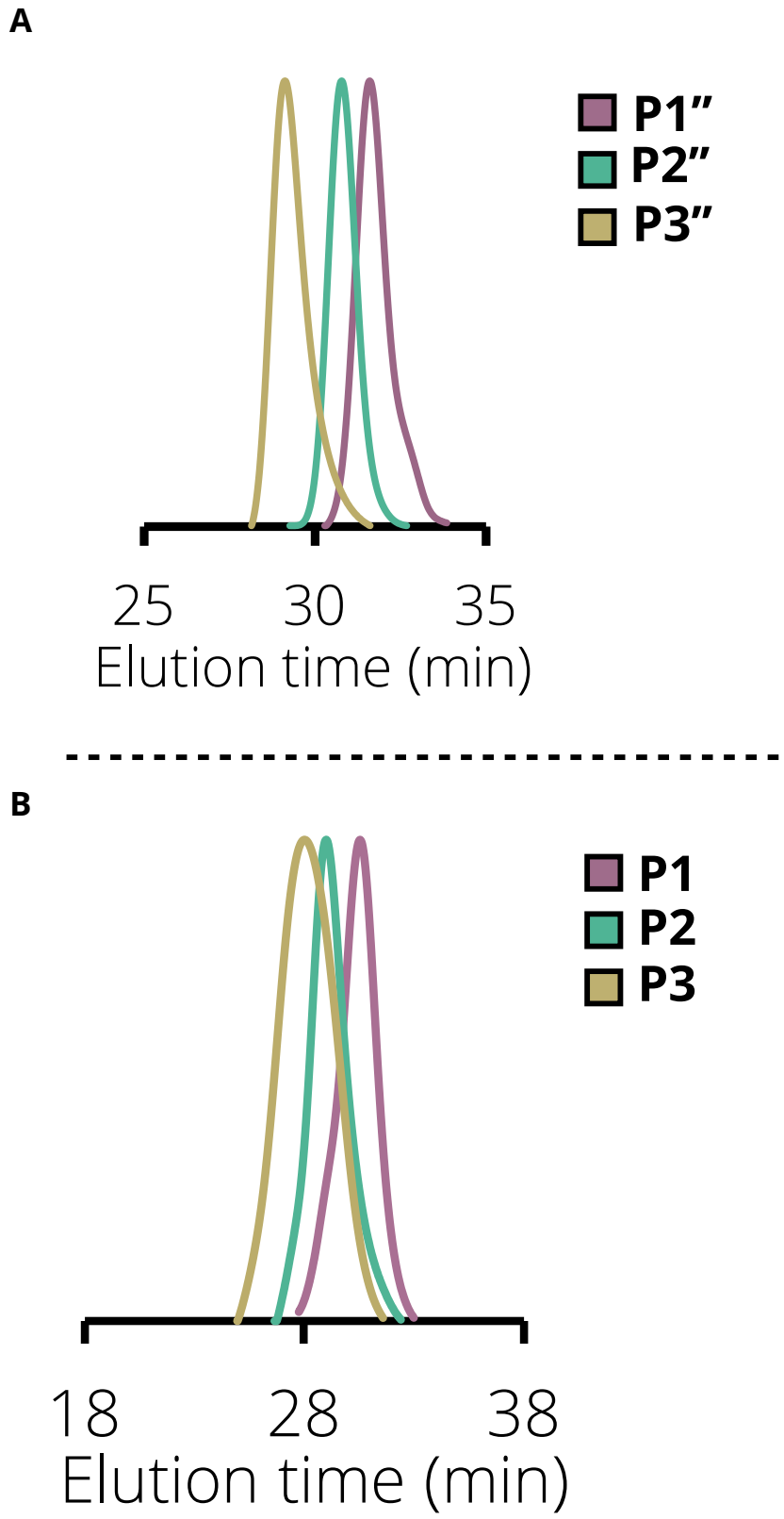


Figure S9. GPC traces of (A) **P1''-P3''** using THF size exclusion chromatography (1 mL/min) and (B) **P1-P3** using aqueous size exclusion chromatography (0.1 M AcOH, 0.02 M glycine, pH 2.3, 0.5 mL/min).

S10. CD spectra of P2

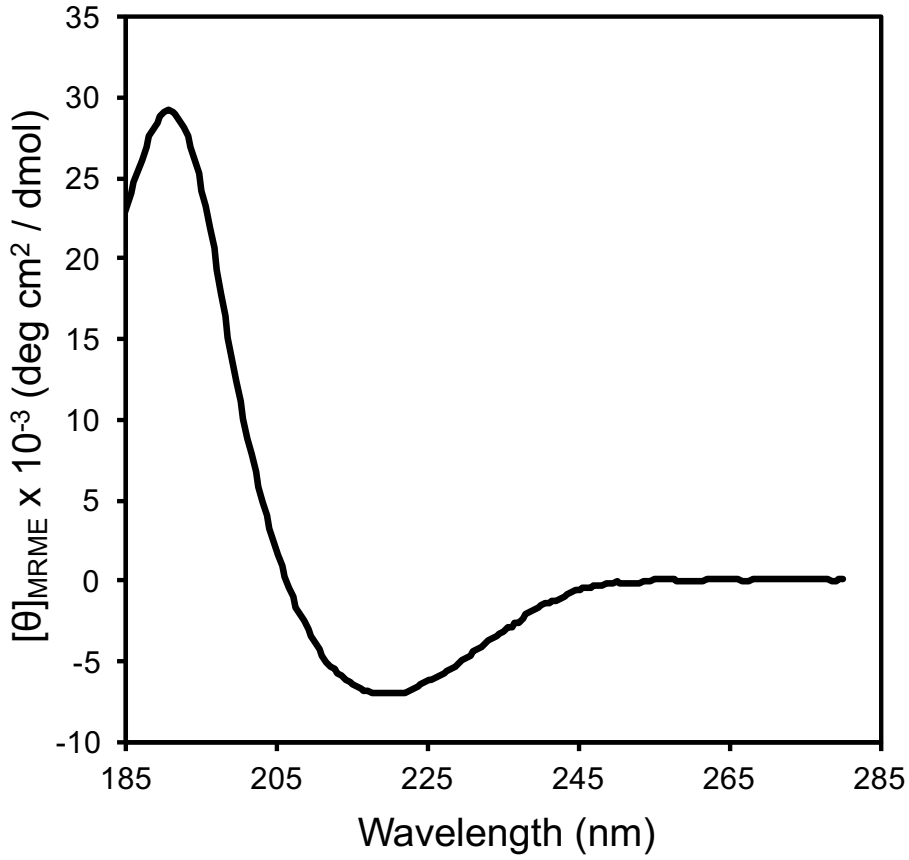
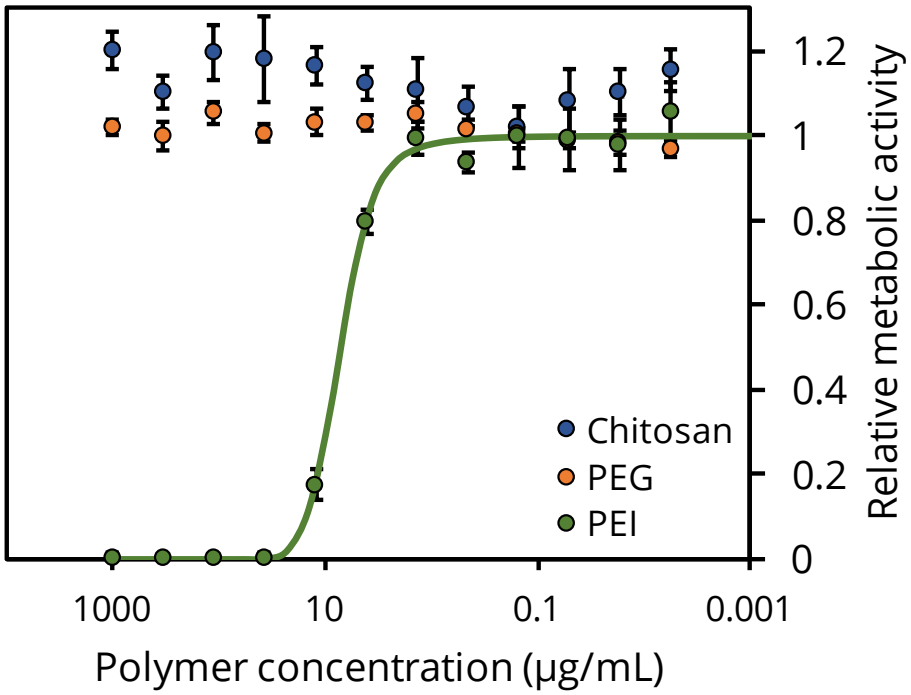


Figure S10. Circular dichroism spectra of **P2** (0.15 mg/mL) obtained in phosphate buffer pH 7.4.

S11. A549 cell proliferation assay with Chitosan, PEG, and PEI



IC_{50} (µg/mL)	PEG	Chitosan	PEI
A549	N/A	N/A	3.8

Figure S11. MTS cell proliferation assay with A549 cells. Cells were allowed to adhere overnight followed by addition of Chitosan, PEG, or PEI serially diluted 1:2 in F12 media with 10% FBS and antibiotics. Following a 24 hour incubation, the MTS dye was added to each well and color was allowed to develop over 4 hours. Each point is an average of n=6 biological replicates with baseline absorbance subtracted and normalized to the absorbance of untreated cells. Data were fit to Hill functions to extract the IC_{50} value for PEI treatment.

S12. Images of mucin functionalized surface

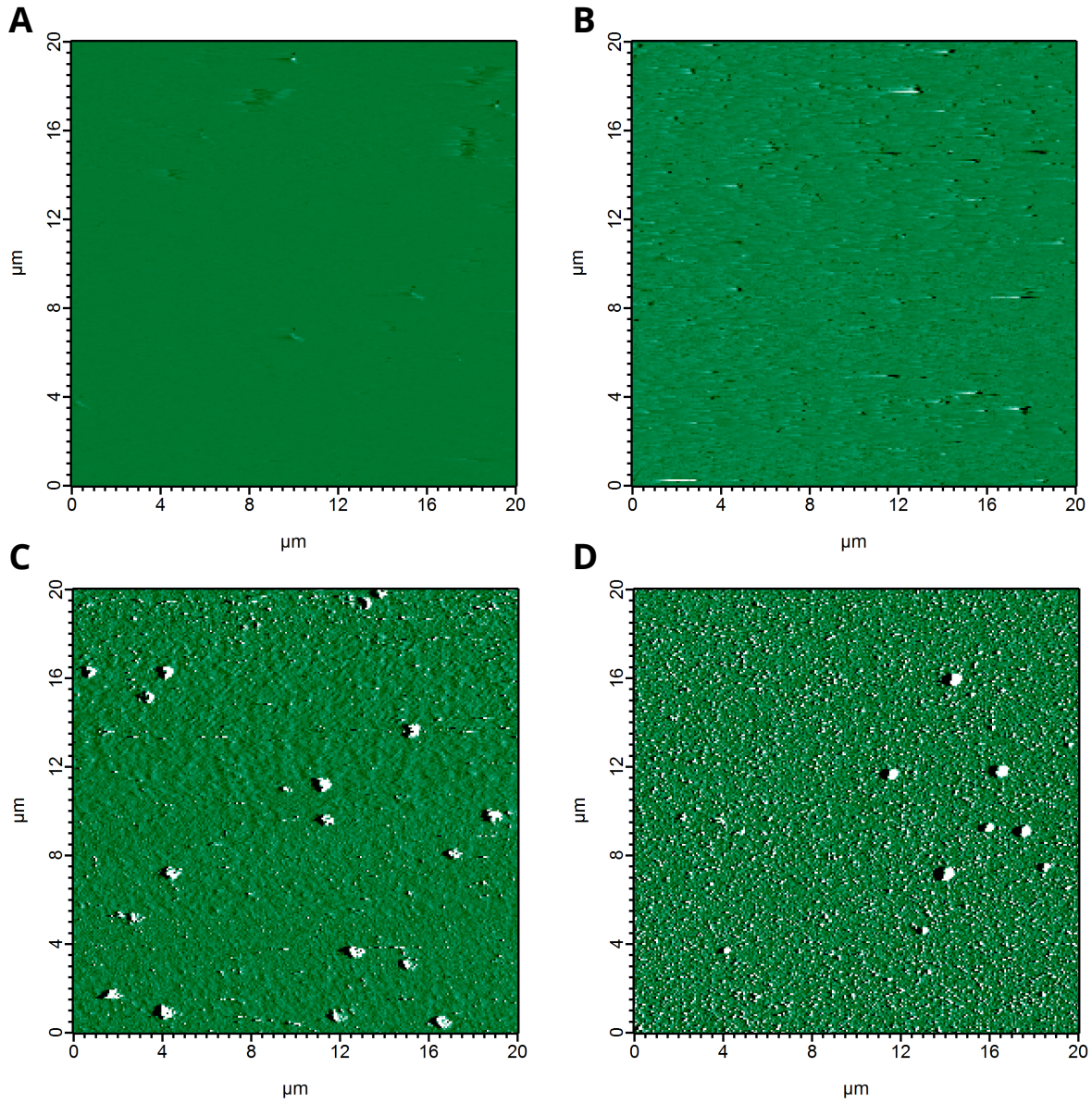


Figure S12. AFM images of tested surfaces (pseudocolor: green lower features, white higher features). (A) Unmodified mica surface; (B) mica surface functionalized with aminosilane groups; (C and D) Two independent mucin functionalized mica discs indicating successful functionalization.

S13. Comparison of adhesion between mucin-functionalized and bare surfaces for all 5 tips

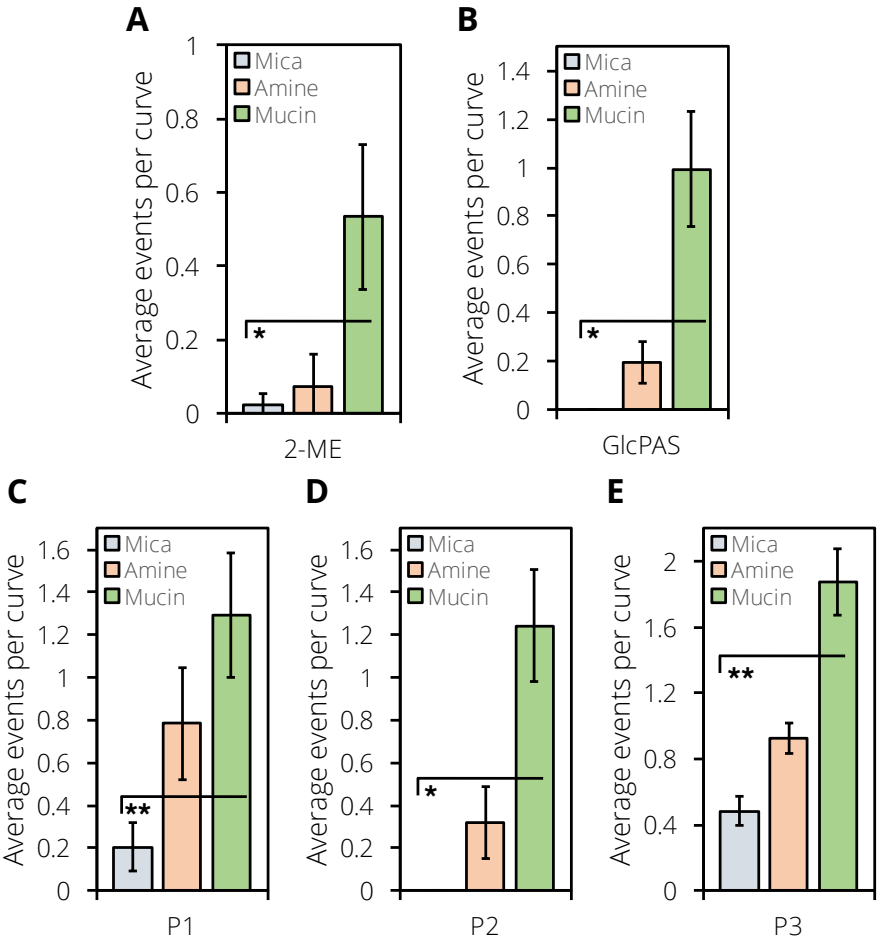


Figure S13. Comparison of adhesion events between mucin functionalized surface and bare surface for all 5 tips. Error bars are average of All tips showed statistically significantly higher adhesion frequency on mucin functionalized surfaces compared to bare surfaces. Data obtained by randomly selecting 5 sets of 20 curves and calculating number of events in each set and obtaining the average and standard deviation. This was repeated three times to tabulate the average number of events per curve and the standard deviations were averaged to generate error bars (2-tailed t-test with unpooled variance; n.s., $p > 0.05$; *, $p < 0.05$; **, $p < 0.01$). (A) 2-mercaptoethanol; (B) GlcPAS; (C) **P1** / AmPAS12SH; (D) **P2** / AmPAS25SH; (E) **P3** / AmPAS50SH.

S14. Table of Mixed Model Parameters

		2ME	GlcPAS	**P1**	**P2**	**P3**
Rupture length (nm)	Mean 1 (%)	87 (100)	55 (65)	58 (64)	70 (72)	76 (53)
	Mean 2 (%)	--	337 (35)	325 (36)	497 (28)	387 (47)
	Weighted average (nm)	87	154	155	190	222
Rupture force (pN	Mean 1 (%)	61 (70)	64 (84)	57 (80)	51 (87)	66 (85)
	Mean 2 (%)	229 (30)	293 (16)	177 (20)	201 (13)	193 (15)
	Weighted average (pN)	111	92	81	71	85
Adhesion energy (aJ)	Mean 1 (%)	0.83 (99)	0.90 (68)	0.96 (77)	0.45 (62)	2.2 (68)
	Mean 2 (%)	28 (1)	13 (32)	22 (23)	22 (37)	33 (32)
	Weighted average (aJ)	1.1	4.8	5.8	8.4	12.1

Table S14. Table of mixed model distribution parameters calculated for rupture lengths, rupture forces, and adhesion energy data for modified tips on mucin-functionalized surfaces.

S15. Rupture length distributions of P1 and P2

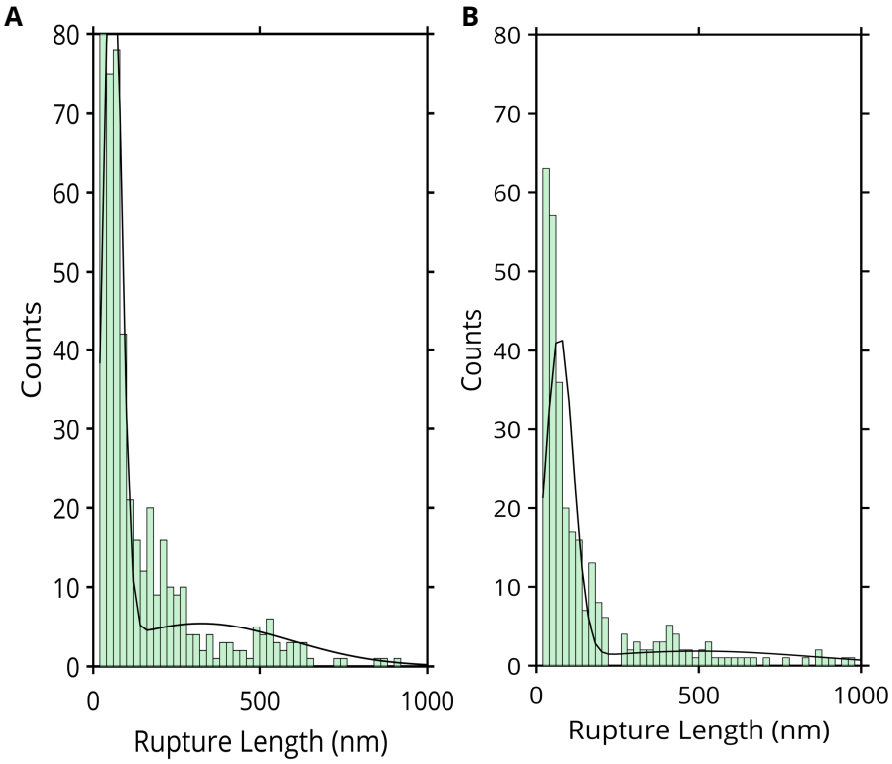


Figure S15. Rupture length distributions of **P1** and **P2** indicate higher number of events and a skew towards events with higher rupture lengths than those observed for 2-ME tips.

S16. Adhesion energy distributions of P1 and P2

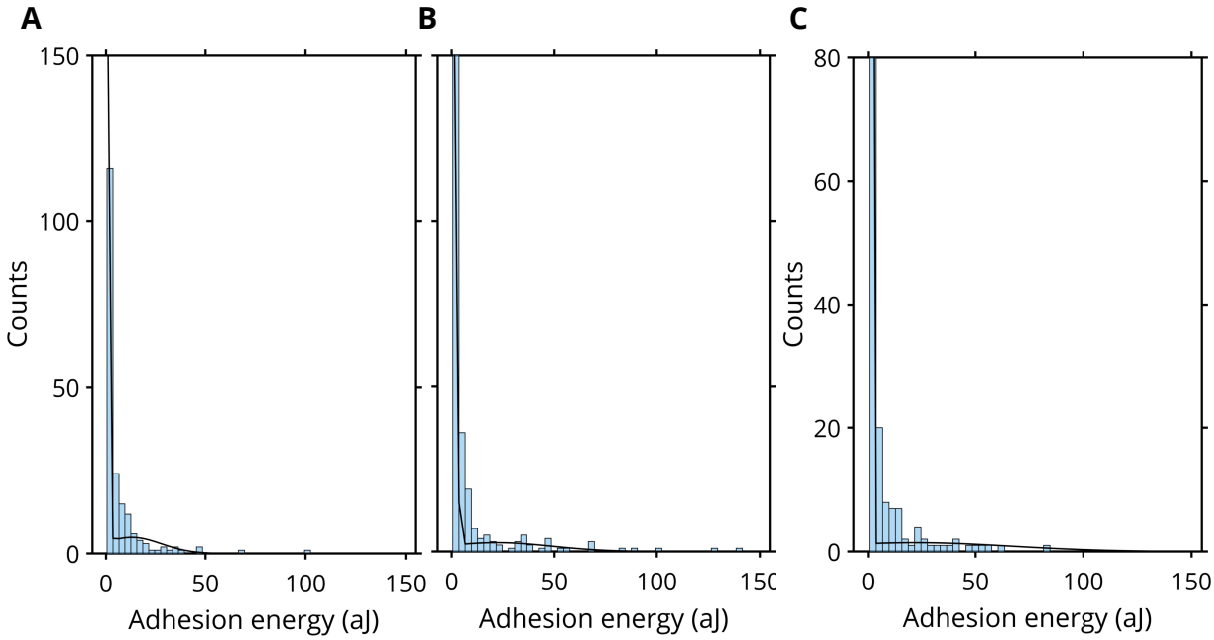


Figure S16. Adhesion energy distributions of P1 (B) and P2 (C) reveal an increased contribution from a distribution with mean of 22 aJ compared to GlcPAS (A) with a mean of 13 aJ. Both P1 and P2 display adhesion events with energies exceeding 40 aJ, while few are observed for GlcPAS.

S17. 200 nm gated interactions and resulting rupture force histograms

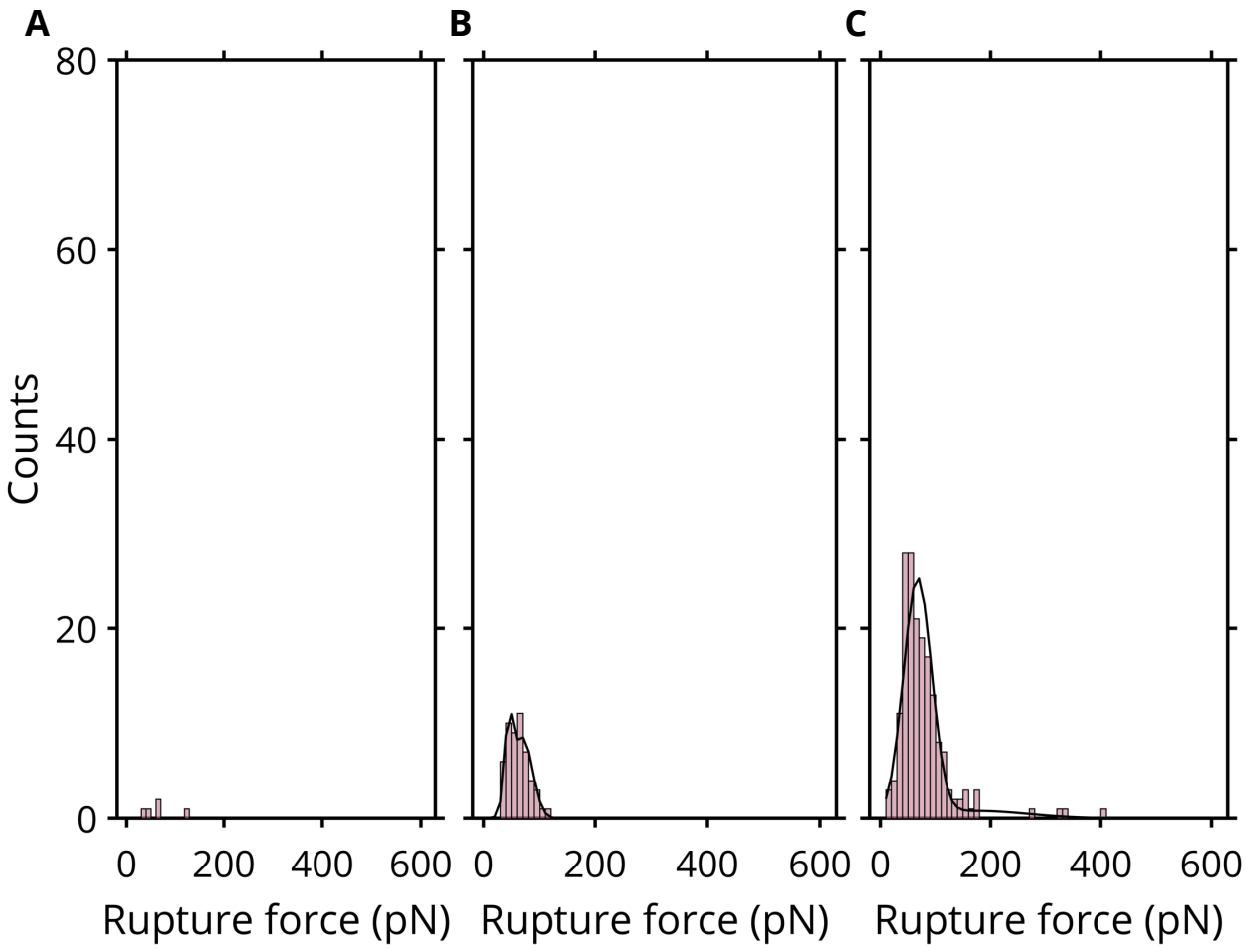


Figure S17. Resulting histograms after application of a 200 nm gate to the rupture force data. (A) 2-ME, few events are observed; (B) GlcPAS, more events are observed with mean rupture force of 70 pN; (C) **P3**, even more events are observed and a distribution with mean rupture force of 200 pN is a significant (10%) contributor.

S18. Fluorescence recovery after photobleaching

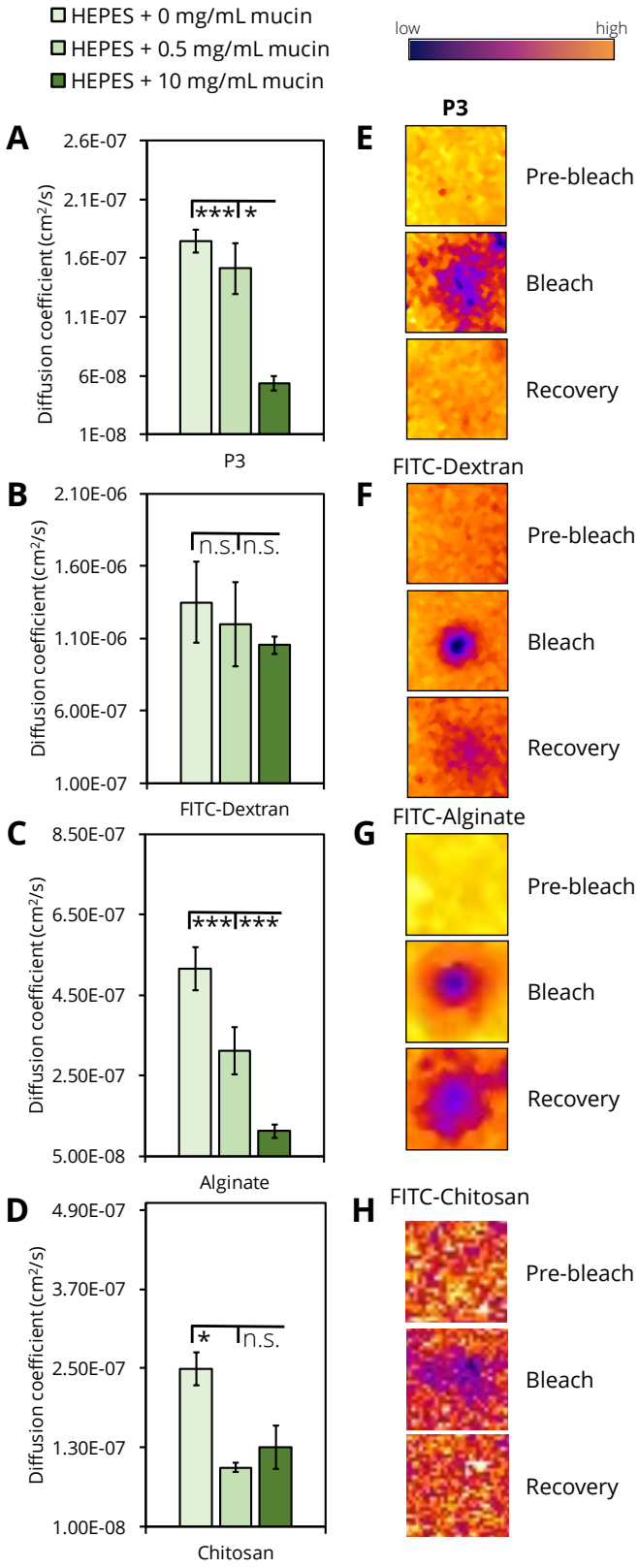


Figure S18. Fluorescence recovery after photobleaching (FRAP) diffusion coefficients (cm2/s) obtained for (A) TAMRA-P3 (B) FITC-dextran (C) FITC-alginate or (D) FITC-chitosan in 10 mM HEPES buffer pH 7.5 with either 0 (light green), 0.5 (medium green), or 10 (dark green) mg/mL of mucin added (n=4). (E-H) Example FRAP images of (E) TAMRA-P3, (F) FITC-dextran, (G) FITC-alginate, or (H) FITC-chitosan taken prior to bleaching, just after bleaching, and after recovery; the high photostability of TAMRA necessitated long bleaching times in order to obtain bleaching >10% for accurate measurement of diffusion coefficient. (n.s., $p > 0.05$; *, $p < 0.05$; **, $p < 0.01$, ***, $p < 0.001$).

S19. Raw FRAP curves

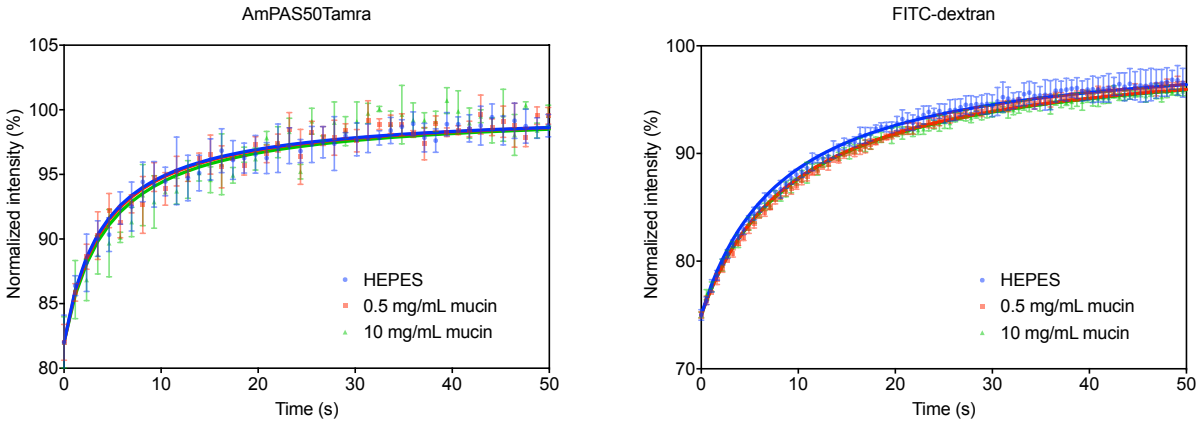


Figure S19. Raw FRAP curves indicating fitting of processed data and subsequent extraction of diffusion coefficients.

Experimental Procedures

Materials and general methods

D-glucal and EDAC were purchased from Carbosynth (San Diego, CA). All other reagents were purchased from Millipore Sigma (Burlington, MA), TCI (Portland, OR), or Chem-Impex (Wood Dale, Illinois). Solvents were purchased from Thermo Fisher Scientific (Waltham, MA). Reactions were monitored by thin-layer chromatography (TLC) analysis, and stained by potassium permanganate or ninhydrin. All ^{1}H-NMR and ^{13}C-NMR spectra were taken with compounds dissolved in $CDCl_3$ [(D, 99.8%) + 0.05% V/V TMS + Silver foil] or CD_3CN (D, 99.8%) and the TMS (0 ppm) was used as an internal standard for ^{1}H-NMR spectra. Chemical shifts (δ) are recorded in ppm, coupling constants (J) are reported in Hz. Unless otherwise noted, all reactions were performed under an argon atmosphere using anhydrous solvents and oven-dried glassware and stir bars. Lyophilization was performed using a Virtis Benchtop 4K freeze dryer Model 4BT4K2L-105 at -40 ºC. ^{1}H NMR and ^{13}C NMR spectra of all small molecules**, P1''-P3''**, and **P1** were recorded on a Varian INOVA 500 MHz spectrometer. ^{1}H NMR and ^{13}C NMR spectra of **P2** and **P3** (at 25 ºC) were obtained on a Bruker AVANCE III HD 500 MHz spectrometer equipped with a cryogenically cooled multinuclear probe tuned to ^{13}C with ^{1}H decoupling. Infrared spectroscopy (IR) was performed on a Nicolet FT-IR with a horizontal attenuated total reflectance (ATR) adapter plate. Circular dichroism (CD) studies were performed in a 1 mm path length cuvette using an Applied Photophysics CS/2Chirascan with a standard Mercury lamp. Polymers were dissolved in water (2 mg/mL) and diluted in the appropriate mixture of buffer and water to a final concentration of 0.15 mg/mL for CD analysis.

Synthetic methods

*(**2**) 6-O-triisopropylsilyl-D-glucal* was prepared as previously described[1]. The crude product was purified by silica column chromatography (4:1 hexanes:ethyl acetate to 1:1 hexanes:ethyl acetate) and the combined fractions were evaporated to dryness to yield 6-O-triisopropylsilyl-D-glucal as a viscous, clear oil (yield 83%). Spectroscopic and mass characterization data matched those previously reported.

*(**3**) 3,4-di-O-benzyl-6-O-triisopropylsilyl-D-glucal* was prepared as previously described[1]. The crude product was purified by silica column chromatography (19:1 hexanes:ethyl acetate to 17:1 hexanes:ethyl acetate) and the combined fractions were evaporated to dryness to yield 3,4-di-O-benzyl-6-O-triisopropylsilyl-D-glucal as a clear oil (yield 63%). Spectroscopic and mass characterization data matched those previously reported.

*(**4**) 3,4-di-O-benzyl-D-glucal* was prepared as previously described[2]. The crude product was purified by silica column chromatography (4:1 hexanes:ethyl acetate to 2:1 hexanes:ethyl acetate) and the combined fractions were evaporated to dryness to yield 3,4-di-O-benzyl-D-glucal as a clear oil that solidified into an amorphous white solid upon storage at -20 ºC (yield 93%). Spectroscopic and mass characterization data matched those previously reported.

*(**5**) 3,4-di-O-benzyl-6-N-(2-nitrobenzenesulfonyl)-N-boc-6-deoxy-D-glucal*. *N-tert*-butoxycarbonyl-2-nitrobenzenesulfonamide was prepared prior to the Mitsunobu reaction by reacting 2-nitrobenzenesulfonamide with 1.1 eq of di-*tert*-butyl-dicarbonate, 1.1 eq of trimethylamine, and 0.1 eq of 4-DMAP in 120 mL of DCM as previously described. The reaction was allowed to proceed for 1 h before the reaction mixture was poured into 500 mL of 1 M HCl and extracted 4 times with 100 mL of diethyl ether. The combined organic extracts were further washed with 1 M HCl, followed by brine and dried over sodium sulfate before evaporating to dryness. The crude solid was repeatedly triturated with 40% diethyl ether in hexanes until the yellow color in the collected washes was eliminated. Compound **4** (5.0 g, 13.8 mmol), was added to 1.0 eq of *N*-boc-2-nitrobenzenesulfonamide (4.2 g, 13.8 mmol) and 1.1 eq of PPh_3 (4.0 g, 15.2 mmol) in 100 mL of THF and the stirring mixture was allowed to cool to 0 ºC. Di-2-methoxyethyl azodicarboxylate (1.1 eq, 3.6 g, 15.2 mmol) was added portionwise at a rate that allowed for immediate removal of yellow color in solution with each addition of solid. The reaction mixture was allowed to warm to room temperature overnight and the solvent was then removed by rotary evaporation. The mixture was dissolved in 300 mL ethyl acetate and washed 3 times with 100 mL of 0.1 M NaOH, brine, dried over sodium sulfate, and evaporated to dryness. The crude product was purified by column chromatography (2:1 toluene:ethyl acetate) and fractions containing the product (permanganate stain, TLC) were collected and evaporated to dryness to yield the title compound (**16**) as a pale, highly sticky and viscous oil (Yield 78%, 6.6 g). $[\alpha]_{25}^{D}$ = +1.0 (0.7, CH_3OH). ^{1}H NMR (500 MHz, Chloroform-d) δ 8.34 (dt, J = 6.2, 2.3 Hz, 1H, Nosyl), 7.79 – 7.66 (m, 3H, Nosyl), 7.34 (m, 10H, Ar), 6.51 – 6.46 (m, 1H, H1), 5.03 – 4.92 (m, 1H, H2), 4.79 (d, J = 11.8 Hz, 1H, ArC*H*$_2$), 4.72 – 4.65 (m, 2H, ArC*H*$_2$), 4.52 (d, J = 11.6 Hz, 1H, ArC*H*$_2$), 4.46 – 4.35 (m, 2H, H3+H6-1), 4.09 (t, J = 4.0 Hz, 1H, H4), 4.02 – 3.94 (m, 1H, H6-2), 3.70 (t, J = 4.7 Hz, 1H, H5), 1.32 (s, 9H, Boc). ^{13}C NMR (126 MHz, Chloroform-d) δ 150.21 (Boc*C*O), 147.71 (Nosyl), 144.39 (C1), 138.23-124.38 (Aryl+Nosyl), 99.14 (C2), 85.11, 74.80 (C5), 74.50 (C3), 72.53 (Ar*C*H$_2$), 72.18 (C4), 69.82 (Ar*C*H$_2$), 47.62 (C6), 27.92 (Boc). HRMS ESI+ TOF (*m/z*): $[M+Na]^+$ calcd for $C_{31}H_{34}N_2O_9S$: 633.1877; found, 633.1868.

*(**6**) 1,2-β-lactam-3,4-di-O-benzyl-6-N-(2-nitrobenzenesulfonyl)-N-boc-6-deoxy-D-glucal*. Compound **5** (6.1 g, 10 mmol), was dissolved to 1 M in MeCN (10 mL) and cooled to 0 ºC. Trichloroacetyl isocyanate (4 eq, 4.9 mL, 40 mmol) was added dropwise and the reaction mixture was allowed to warm to room temperature. The reaction was monitored from 7-10 days by TLC (permanganate stain), or until the starting material was completely consumed. The reaction mixture was then cooled to -20 ºC with a water/methanol cooling bath, followed by the addition of 1.1 eq (to TCAI) of benzylamine (4.8 mL, 43.9 mmol) dissolved in 20 mL of acetonitrile dropwise by syringe pump over 1 hour. The reaction mixture was then allowed to warm to room temperature while monitoring; once its temperature reached ~0 ºC, a white precipitate was formed in a yellow solution. The mixture was filtered to remove the white

precipitate and the flask and filtered solid were quickly triturated with hexanes before the residual acetonitrile could evaporate. The filtrate was then poured into 600 mL of 0.1 M HCl and extracted 4 times with ethyl acetate. The combined organic extracts were washed 2 times with 0.1 M HCl, brine, dried over sodium sulfate, and evaporated to dryness. The crude product was purified by column chromatography (2:1 hexanes:ethyl acetate to 1.5:1 hexanes:ethyl acetate to 1:1 hexanes:ethyl acetate) and fractions containing the product (permanganate stain, TLC) were collected and evaporated to dryness to yield the title compound (**17**) as a pale foam. (Yield 38%, 2.5 g). ^{1}H NMR (500 MHz, Chloroform-d) δ 8.35 – 8.27 (m, 1H, Nosyl), 7.79 – 7.69 (m, 3H, Nosyl), 7.46 – 7.21 (m, 10H, Ar), 6.14 (d, J = 2.4 Hz, 1H, N*H*), 5.57 (d, J = 4.4 Hz, 1H, C1), 4.79 (d, J = 11.6 Hz, 1H, ArCH_2), 4.69 (d, J = 11.8 Hz, 1H, ArCH_2), 4.59 (dd, J = 28.1, 11.6 Hz, 2H, ArCH_2), 4.25 –4.07 (m, 3H, H4, H6-1, H3), 4.04 – 3.83 (m, 1H, H6-2), 3.53 – 3.42 (m, 2H, H5+H2), 1.30 (s, 9H, Boc). ^{13}C NMR (126 MHz, Chloroform-d) δ 166.88 (Lactam *C*O), 150.14 (Boc), 137.75-124.43 (Ar+Nosyl), 77.22 (C5), 76.18 (C3), 75.98 (C1), 73.14 (Ar*C*H$_2$), 71.17 (Ar*C*H$_2$), 68.24 (C4), 54.78 (C2), 49.00 (C6), 27.82 (Boc). LC/MS (*m/z*): [M+Na]$^+$ calcd for $C_{32}H_{35}N_3O_{10}S$: 676.2; found, 676.3.

*(**7**) 1,2-β-lactam-3,4-di-O-benzyl-6-N-boc-6-deoxy-D-glucal*. Compound **6** (3.40 g, 5.2 mmol), was dissolved to 0.2 M in MeCN and 1.2 eq of PhSh was added (640 µL, 6.2 mmol). The reaction mixture was cooled to 0 ºC and 2.2 eq of K_2CO_3 (1.58 g, 11.4 mmol) was added. The reaction mixture was allowed to stir overnight and a strong orange color developed, followed by a yellow color. The mixture was filtered and the solvent was removed by rotary evaporation. The residue was then dissolved in ethyl acetate and washed three times with 0.1 M HCl, brine, dried over sodium sulfate, and evaporated to dryness. The crude product was purified by column chromatography (95:4:1 dichloromethane:acetone:methanol to 93:4:3 dichloromethane:acetone:methanol) and fractions containing the product (permanganate stain, TLC) were collected and evaporated to dryness to afford the title compound (**20**) as a white foam. (Yield 75%, 1.83 g). $[\alpha]_{25}^{D}$ = +3.4 (0.4, CH_3OH). ^{1}H NMR (500 MHz, Acetonitrile-d3) δ 7.60 – 7.19 (m, 10H, Ar), 6.70 (s, 1H, N*H*), 5.44 (d, J = 4.5 Hz, 1H, H1), 5.30 (s, 1H, N*H*Boc), 4.68 (dd, J = 19.8, 11.4 Hz, 2H, ArCH_2), 4.54 (t, J = 11.0 Hz, 2H, ArCH_2), 4.00 (dd, J = 5.3, 2.7 Hz, 1H, H3), 3.89 (q, J = 6.5, 5.7 Hz, 1H, H4), 3.52 – 3.48 (m, 1H, H2), 3.44 (dd, J = 8.1, 5.4 Hz, 1H, H5), 3.41 – 3.23 (m, 2H, H6), 1.42 (s, 9H, Boc). ^{13}C NMR (126 MHz, Acetonitrile-d3) δ 166.61 (Lactam *C*O), 138.38 (Boc *C*O), 128.73 – 127.16 (Ar), 76.63 (C5), 75.60 (C1), 75.37 (C3), 72.57 (Ar*C*H$_2$), 70.43 (Ar*C*H$_2$), 68.93 (C2), 53.55 (C6), 27.62 (Boc). HRMS ESI+ TOF (*m/z*): [M+Na]$^+$ calcd for $C_{26}H_{32}N_2O_6$: 491.2153; found, 491.2157.

*(**9**) Pentafluorophenyl S-benzylthioglycolate*. Compound **8** (5.0 g, 27.4 mmol) was dissolved to ~0.25 M in 100 mL of DCM. To this stirring mixture was added 1.1 eq (4.2 mL, 30.2 mmol) of Et_3N, 1.1 eq of NHS (3.5 g, 30.2 mmol), 0.1 eq (330 mg, 2.7 mmol) of 4-DMAP, and 1.1 eq (5.6 g, 30.2 mmol) of pentafluorophenol. Each of these components was allowed to dissolve completely and the mixture was then cooled to 0 ºC. EDAC (1.1 eq, 4.7 g, 30.2 mmol) was added in 3 separate portions over 30 minutes. The reaction mixture was allowed to warm to room temperature overnight and was then diluted to 200 mL of DCM. The mixture was then washed 3 times with 1 M HCl, 2 times with 0.1 M NaOH (quickly), brine, dried over sodium sulfate, and evaporated to dryness. The crude mixture was purified by column chromatography (6:1 hexanes:ethyl acetate, blue fluorescence under short-wave UV) and fractions containing the product were collected and evaporated to dryness. The product was obtained as a brown oil that solidified upon cooling to -20 ºC. (Yield 81%, 7.7 g). ^{1}H NMR (500 MHz, Chloroform-*d*) δ 7.40 – 7.24 (m, 5H, CH2*Ar*), 3.90 (s, 2H, OC*H2*Ar), 3.35 (s, 2H, SC*H2*C). ^{13}C NMR (126 MHz, Chloroform-*d*) δ 166.31 (*C*OO) , 142.16 (Pfp), 142.05 (Pfp), 136.94 (Pfp), 136.34, 129.19 (Ar), 128.71 (Ar), 127.58 (Ar), 36.13 (O*C*H2Ar) , 31.04 (S*C*H2C). ^{19}F NMR (470 MHz, Chloroform-*d*) δ -152.55 (m, 2F), -157.47 (t, *J* = 21.6 Hz, 1F), -162.00 (m, 2F). HRMS ESI+ Q-TOF (*m/z*): [M+Na]+ calcd for C15H9F5O2S : 348.0243; found, 348.0281.

*Synthesis of **P1''-P3''***

Freshly distilled THF was degassed with argon for 30 minutes while simultaneously evacuating and re-filling a Schlenk flask loaded with 3 Å molecular sieves with argon. LiHMDS was obtained from a glove box and a solution was prepared using the freshly-degassed THF (solution was stable for >12 h with no color / oxidation visible). A solution of initiator (**9**) was prepared using this freshly-degassed THF. Monomer **7** was weighed out (200 mg, 0.427 mmol), dissolved in 3 mL of degassed THF, and added to the Schlenk flask. The reaction mixture was cooled to 0 ºC and the appropriate amount of initiator (2 mol % for 50-mer, 4 mol % for 25-mer, 8 mol % for 12-mer) was added (calculated to be 500 µL) via a glass syringe to the Schlenk flask. After 10 minutes, 2.5 eq of LiHMDS to the initiator was added (also calculated to be 500 µL for ease of measurement) via glass syringe to the Schlenk flask quickly. The reaction mixture developed a yellow color and, after 30 minutes the ice bath was removed to allow the reaction to warm to room temperature. The reaction was then monitored by TLC for consumption of the starting material (permanganate stain, UV, TLC). The solvent was removed from the reaction mixture by rotary evaporation and the residue was dissolved in DCM, washed with 1 M HCl, saturated bicarbonate, brine, dried over sodium sulfate, and evaporated to dryness. The crude residue was dissolved in a minimum amount of DCM and added dropwise to a stirring, ice-cold solution of pentane. The precipitate was obtained via filtration and air-dried followed by vacuuming drying to yield **P1''** as a white powder. (Yields 75-82%, 150-164 mg). ^{1}H NMR (500 MHz, $CDCl_3$) δ 8.21-7.80 (br), 7.55-6.55 (br), 6.21-5.10 (br), 5.08-3.88 (br), 3.88-2.46 (br), 1.63-0.96 (br). SEC (THF) M_w = 6.8 kDa, M_n = 6.0 kDa, Đ = 1.1.

*Synthesis of **P1-P3***

Polymers **P1''-P3''** (150 mg, 0.32 mmol repeating unit) were dissolved in 4 mL of DCM and 2 mL of TFA was added to the stirring solution (due to TFA fumes, the reactions were conducted in glass vials and covered with glass stoppers). The reaction was allowed to proceed for 4 hours and the solvent was then removed by rotary evaporation. The residual TFA was removed by repeatedly adding DCM, dissolving the contents of the flask, and evaporating the mixture. After 3 of these rounds of evaporation, a sticky solid residue was obtained. To this residue was added 2 mL of 1 M NaOH and 2 mL of DCM and the mixture was vigorously dissolved with vortexing. More 1 M NaOH and DCM was added in a 1:1 ratio until the total solution volume reached 8 mL to aid in dissolving the crude. Then, the mixture was diluted with 8 mL of DCM and washed 4 times with 4 mL of 0.1 M NaOH. The combined aqueous

washes were back-extracted with 4 mL of DCM and this was added to the organic extracts. 4 mL of a 0.1 M NaOH / brine solution was added to the vial and, after washing, the organic layer was transferred to a fresh vial and sodium sulfate was added. The mixture was then filtered (and previous vials rinsed thoroughly with DCM) and the solvent was removed under rotary evaporation. A 250 mL tri-neck flask was equipped with a glass stir bar and condenser and was cooled to -63 °C (condenser was cooled to -78 °C). The flask was connected to an anhydrous NH_3 tank and 100 mL of ammonia was allowed to condense over 30 minutes while stirring. While ammonia was condensing, freshly distilled THF was degassed with argon for 30 minutes while simultaneously aliquoting LiHMDS from a glove box. The crude residue was then dissolved in 2 mL of freshly degassed THF in a vial equipped with a stir bar, while LiHMDS was dissolved to form > 1 mL of a 1 M solution. Sodium metal was cut in cyclohexane and added to the liquid ammonia to generate a deep blue-colored solution, followed by the addition of 3.0 eq of LiHMDS (vs. repeating unit, 960 µL, 0.960 mmol) to the THF solution of crude residue with vigorous stirring. The mixture of LiHMDS and crude product formed a bright red color, and the solution was transferred dropwise to the flask with liquid ammonia. The reaction was allowed to proceed for 45 ($[M]_o/[I]$ = 12), 60 ($[M]_o/[I]$ = 25), 90 ($[M]_o/[I]$ = 50), or 120 minutes ($[M]_o/[I]$ = 100) and a saturated solution of NH_4Cl was added dropwise to quench the reaction. The resulting clear solution was allowed to evaporate overnight while stirring to yield a white solid or a white paste. This white residue was dissolved in water and extracted twice with diethyl ether. The aqueous layer was then added to 1 kDa MWCO dialysis tubing and dialyzed overnight (changes at 4, 8, and 16 h). The contents of the dialysis bag were recovered after 24 h, passed through a 0.45 µm syringe filter, and lyophilized over 3 days to obtain **P1**-**3** as a white, cotton-like, fluffy solid. (Yield 48-73% over two steps, 39 mg **P1**, 56 mg **P2**, 59 mg **P3**). ^{1}H NMR (500 MHz, D_2O) δ 5.68-5.38 (bm, 1H, H1), 3.98-3.60 (bm, 1H, H4), 3.46-3.25 (bm, 1H, H3), 3.21-2.66 (bm, 3H, H5, H6-1, H6-2). ^{13}C NMR (126 MHz, $CDCl_3$) δ 169.43 (NH*C*O), 74.40 (C1), 71.83 (C5), 76.14 (C3), 69.80 (C4), 51.49 (C2), 40.34 (C6). SEC (H_2O, 0.1 M NaOH, 0.02 M glycine, pH 2.3) M_w = 4.7 kDa, M_n = 4.1 kDa, Đ = 1.1.

Size exclusion chromatography

THF Protected polymer molecular weights were determined by size exclusion chromatography (SEC) versus polystyrene standards using THF as the eluent at a flow rate of 1.0 mL/min through two Jordi columns (Jordi Gel DVB 105 Å and Jordi Gel DVB 104 Å, 7.8 x 300 mm) at 25 °C in series with a refractive index detector. All calculations were performed using Breeze GPC software (Waters, Milford, MA). Deprotected polymer molecular weights were determined by aqueous GPC versus poly(2-vinylpyridine) standards using aqueous buffer (0.1 M AcOH, 0.02 M glycine, pH 2.3) as the eluent at a flow rate of 0.5 mL/min through two PL aquagel columns (OH 60 micron, 7.8 x 300 mm) at 25 °C in series with a refractive index detector. All calculations were performed using Cirrus GPC software (Agilent, Santa Clara, CA).

Cell culture methods and MTS experiments

A549, BEAS-2B, and HLF-1 were obtained from ATCC (Manassas, VA) and all experiments were conducted on cells between P3 and P10. A549 and HLF-1 cells were cultured in Ham's F12K media (Invitrogen, Carlsbad, CA) supplemented with 10% FBS and 1% Penicillin / Streptomycin in a 37 °C incubator with 5% CO_2. BEAS-2B cells were cultured using BEGM media (Lonza, Morristown, NJ) supplemented with the BEGM bullet kit and 1% Penicillin / Streptomycin in a 37 °C incubator with 5% CO_2. All cells were passaged upon reaching 70-90% confluency. For MTS experiments, cells were plated to 50% confluency on 96 well plates and allowed to adhere overnight. Then, the media was removed and 150 µL of a 1 mg/mL solution of polymer (**P1**-**P3**, Chitosan, PEG, or PEI) in the appropriate media was added to column 1 and 100 µL of media alone was added to columns 2-12. 50 µL of the solution from column 1 was serially diluted (1:2 dilution) across all columns and the final 50 µL was discarded. The cells were incubated in these polymer solutions for 24 hours, followed by aspiration of the media, rinsing with PBS, and the addition of a 9:1 solution of media to MTS reagent (Promega, Madison, WI). The plates were then loaded onto a plate reader performing kinetic absorbance measurements of the plate until the maximum absorbance reached 1.0. Cells exposed to media alone (n=12) were considered 100% proliferation activity, while wells containing no cells (n=12) were considered 0% proliferation activity and the absorbance of each experimental well (n=6 to n=9) was normalized to this range.

Mouse cytokine array panel

RAW 264.7 cells were cultured in RPMI 1640 media (Invitrogen, Carlsbad, CA) supplemented with 10% FBS and 1% Penicillin / Streptomycin in a 37 °C incubator with 5% CO_2 and used in experiments between P7 and P10. Cells were plated to 70% confluency on 24 well plates and allowed to adhere overnight. Then, the media was aspirated and replaced with either media (n=6), media containing 10 µg/mL **P1**, media containing 10 µg/mL **P3**, or 20 ng/mL LPS. After 24 h, the media was collected for analysis using the Mouse Cytokine Array Kit, Panel A (R&D Systems, Minneapolis, MN) as previously described[3]. Briefly, this assay (see Figure E1) is utilizes nitrocellulose membranes that have been pre-spotted and immobilized in duplicate with various (carefully selected) capture antibodies. The collected media / cell culture supernatants are then incubated (in blocking buffer) with biotinylated detection antibodies to bind to selected cytokines. The mixture, including these cytokine/antibody complexes, are transferred to the blocked nitrocellulose membrane loaded with capture antibodies. The capture antibodies bind the cytokine/antibody complexes to immobilize them onto the membrane. Following washing steps to remove unbound protein and antibodies, Streptavidin-Horseradish Peroxidase (SA-HRP) is added to bind specifically to the biotinylated detection antibodies. After more washing steps to remove unbound SA-HRP, the chemiluminescent substrate is added in order to generate a signal that can be detected by an imaging instrument, such as a BioRad (Hercules, CA) GelDoc. The amount of signal is proportional to the amount of bound biotinylated detection antibody, which is proportional to the amount of cytokine successfully captured.

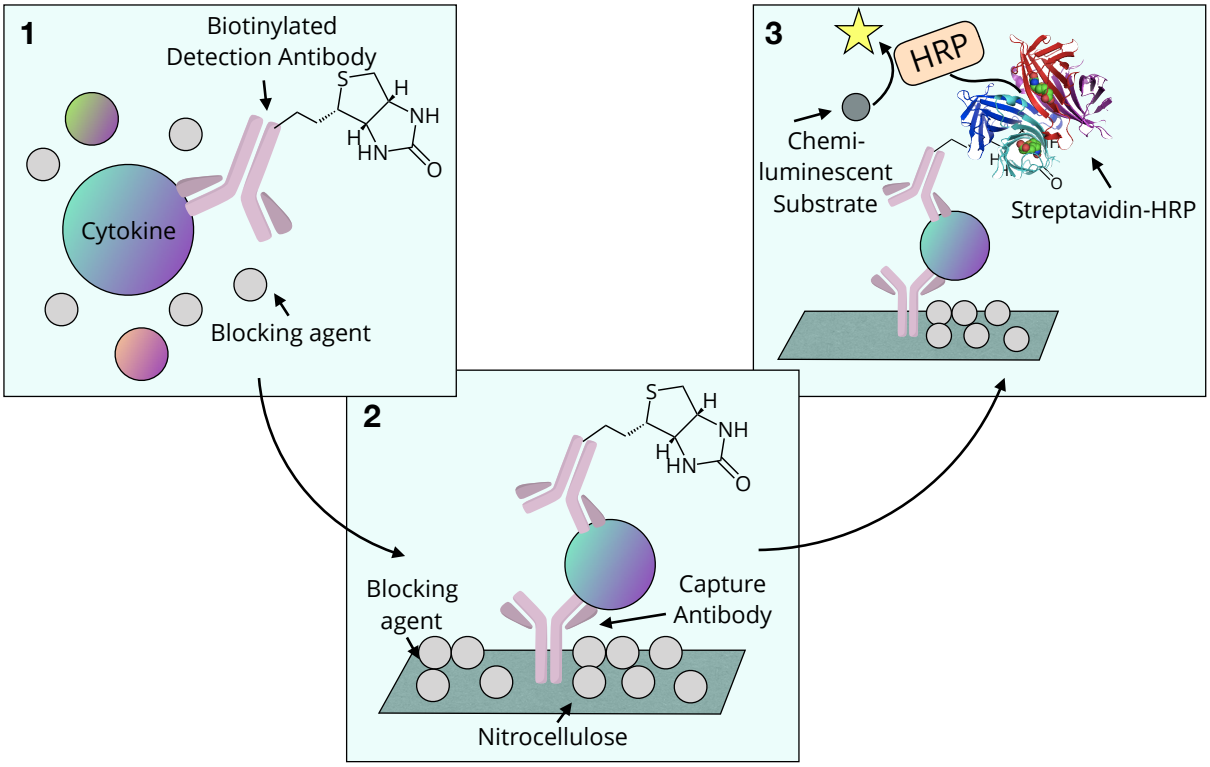


Figure E1. Assay principle of Cytokine Array Panel A. (1) Cell culture supernatants are incubated with biotinylated detection antibodies in blocking buffer. (2) After binding, these antibody/antigen complexes are transferred to a nitrocellulose membrane where a capture antibody immobilizes the antigen onto the blocked membrane surface with the biotinylated antibody still attached. (3) Streptavidin-Horseradish peroxidase is added and it binds the biotinylated capture antibody. (4) A chemiluminescent substrate is added to develop a signal that is measured using a GelDoc (BioRad, Hercules, CA).

Single Molecule Force Spectroscopy

Sample preparation

Mica discs (Ted Pella, Redding, CA) were freshly cleaved and immediately placed into a plastic petri dish with a bath of 3 mL of 1:1 MeOH:HCl (9M) and cleaned for 30 minutes. Then, the discs were rinsed with ddH_2O 3 times (and one set aside as the bare mica surface) before being placed into plastic petri dishes each with 3 mL of 1% (v/v) *N*-[3-(Trimethoxysilyl)propyl]ethylenediamine in 5 mM acetic acid, passed through a 0.45 µm syringe filter. During a 1 hour incubation, a solution of porcine gastric mucin (Millipore Sigma, Burlington, MA) (10 mg/mL) was prepared in 1 mL (per mica disc) of 1 mM MES buffer pH 6.0 with the addition of 0.5 mL of a 0.1 mg/mL solution of EDAC in 1 mM MES buffer pH 6.0. After the 1 hour incubation, the discs were again rinsed 3 times with ddH_2O (and one set aside as the amine silanized surface) and placed into another plastic petri dish (with care not to touch the circular face). Then, the mucin/EDAC solution was added to the petri dishes, as well as 1.5 mL of 100 mM borate buffer, pH 8.0. The reaction was allowed to proceed overnight with gentle shaking. A fresh pack of 10 gold-coated NPG-10 AFM tips (Bruker Nano, Camarillo, CA) was opened while simultaneously bringing 100 mL of 100% ethanol to boiling. The tips were cleaned using O_2 plasma (5 minutes, 100 mW) and immediately placed into a 24-well plate filled with 500 µL of boiling ethanol. After 20 minutes, the tips were washed by sequentially dipping into 3 wells and placed in 500 µL of a 1 mg/mL aqueous solution of the appropriate polymer (or 2-mercaptoethanol) with 1 mg/mL tris(hydroxypropyl)phosphine. The thiol-gold interaction was allowed to equilibrate overnight. On the day of the experiment, the mucin-functionalized mica discs were then rinsed with 10 mM PBS pH 7.4, washed with a PBST (0.05% Tween) solution for 5 minutes (3 total washes), and rinsed with PBS again 3 times. The mucin-functionalized mica surfaces were kept immersed in PBS until measurement. The functionalized AFM tips were next moved to a well containing 500 µL of a 1 mg/mL aqueous solution of 2-mercaptoethanol with 1 mg/mL of tris(hydroxypropyl)phosphine to fill in regions of the gold surface that had not been conjugated. After a 30 minute incubation, the tips were washed by sequential dipping into 3 wells with 10 mM PBS pH 7.4 0.05% Tween, and then 3 wells with PBS alone. The tips were left in PBS until measurement.

Data collection on instrument

Atomic force microscopy (AFM) was performed using a MFP-3D microscope (Asylum Research; Santa Barbara, CA). Contact mode was applied using either a silicon nitride tip with a nominal spring constant of 40 pN/nm, or a gold coated tip (MLCT, Bruker AFM Probes; Camarillo, CA) with a nominal spring constant of either 60 or 120 pN/nm. The tested surfaces (bare mica, amine silanized, mucin functionalized) were fixed onto a glass slide with glue and quickly immersed in 100 µL of PBS. The appropriate tip was slowly

lowered to the surface to confirm contact without damage. Tip calibration was done on the bare mice surface, which acts an infinitely hard surface, to set both a baseline deflection and virtual deflection correction, and determining the cantilever's inverse optical lever sensitivity (InvOLS; unit: m/V). The spring constant was measured on every single measured surface for every single tip via thermal tuning, and was always within the range dictated by Bruker (within 30 pN/nm). After calibration of a particular tip on a particular surface, an area scan was performed in tapping mode prior to indentation to confirm the presence and density of the mucin (images were collected using SiN_3 tips in order to minimize interactions). Areas of mucin functionalization were chosen for indentation based on shape and size. Force spectroscopy measurements were conducted using a 2 μm extension length, a 1000 nm/s tip velocity, and a trigger set-point of 0.75 V of deflection. Once an indentation was performed, the raw distance of the tip along the z-direction was converted into an indentation depth using the InvOLS. Force curves (N=50) were collected over at three different locations on each surface for a total of 150 curves per surface per tip.

Data loading and computational extraction

AFM data was transferred from the software (Igor Pro) to Excel to facilitate extraction from MATLAB, with filenames identifying the batch, surface, and replicate number. After providing the appropriate filename, all curves of each surface and replicate number for each tip were loaded into MATLAB. The curves were then manually inspected to remove those with artifacts, with unmatched retraction / extension dimensions, and with noisy baselines. The pruned data sets were analyzed to identify the x-distance corresponding to the surface (inflection point) by looping from the end of the retraction curve (assumed to be baseline, allowing for the extraction of mean baseline value and average slope) towards the beginning of the retraction curve until the local derivative exceeded the average slope at baseline. This x-distance was set as x=0 nm for all curves. Then, the retraction curve was subtracted from the extension curve to identify regions of hysteresis (adhesion events). The data was analyzed by calculating the global average derivative (window averaged over 10 points to reduce noise) and then looping from the beginning of the subtracted curve to the end while calculating the local derivative. Adhesion events were identified as those displaying local derivatives significantly lower than the global average derivative for at least 3 consecutive points and with a maximum rupture force significantly greater than the baseline. The peak window was identified using a combination of local derivative and local average value, allowing for extraction of peak rupture force, rupture length, adhesion energy, and the event rigidity. Data processing allowed for separation of peaks into individual subpeaks or clustering of subpeaks into 1 single event. The resulting data sets were then compiled by a script that output the appropriate histograms and scatter plots, as desired.

Fluorescence Recovery After Photobleaching (FRAP)

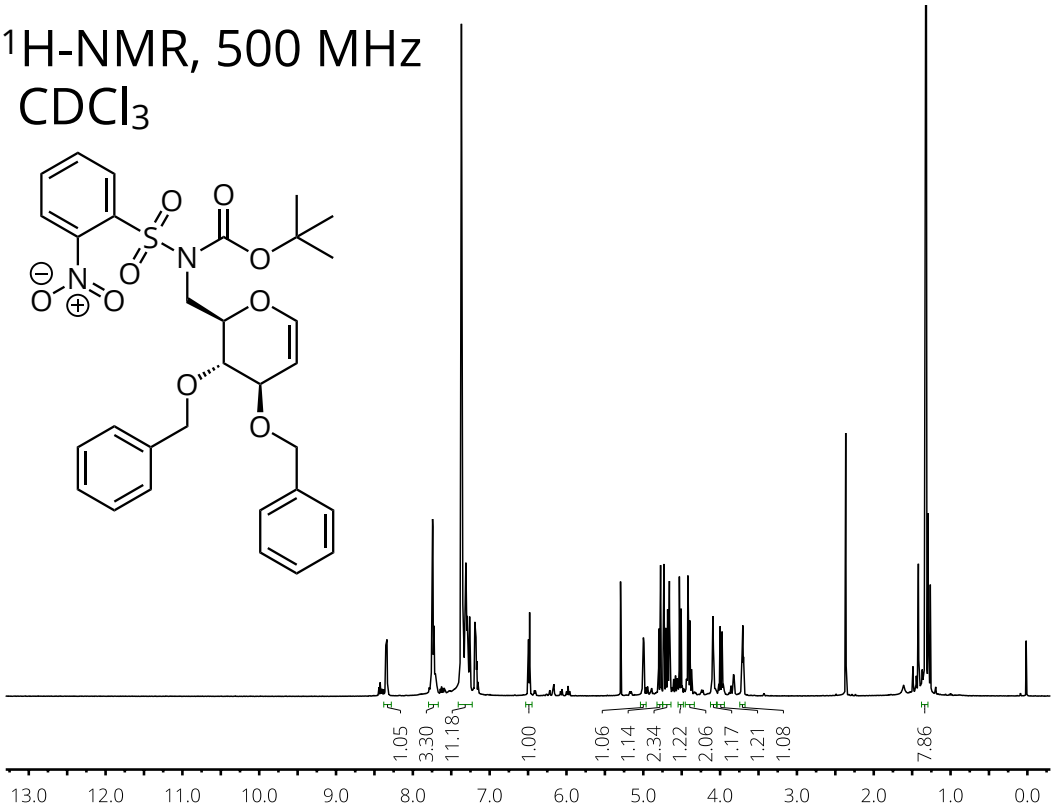
Solutions of 20 mM HEPES pH 7.4 with either 0 mg/mL, 0.5 mg/mL mucin, or 10 mg/mL mucin (0.3 mL) were loaded with 0.3 mg/mL FITC-dextran, TAMRA-AmPAS50, FITC-alginate, or FITC-chitosan respectively, and imaged at 25 °C on an Olympus FV1000 scanning confocal microscope equipped with a 488 nm and 543 nm laser, for FITC and Tamra excitation, respectively. For FITC-dextran, FITC-alginate, and FITC-chitosan, a second (405 nm) scan head supplied the 5 s bleach pulse. Excitation laser power was set to 100%, and fluorescence images were collected. For image acquisition, 100 time series images were obtained (including 5 pre-bleach images) for each molecule. For TAMRA-**P3**, a similar procedure was performed except that bleaching was achieved with 405,458,488,515 and 543 nm lasers for 60 s. Diffusion coefficients were calculated by importing images into MATLAB and calculating the average intensity as a function of time within the ROI. The ROI average intensity *versus* time was fit to a Gaussian model (step size = 0.001), according to the equations given by Jain et al[4], to arrive at the diffusion coefficient, *D*, in cm^2/sec.

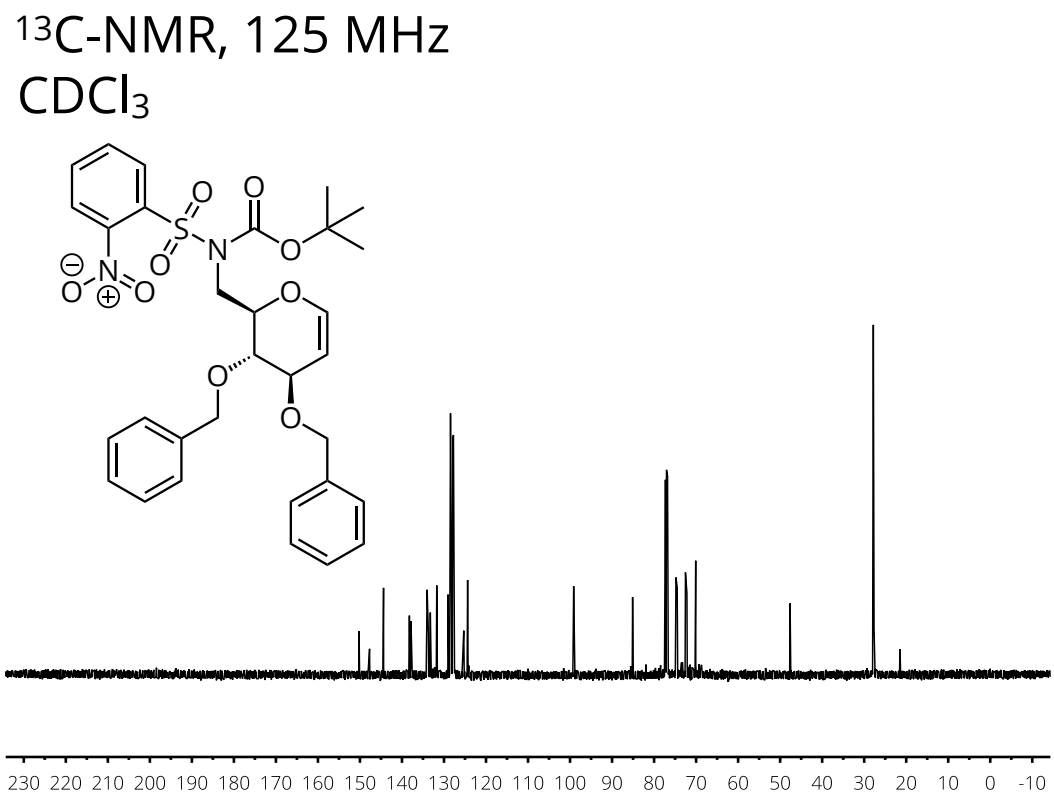
Ex vivo tissue adhesion experiment and quantification

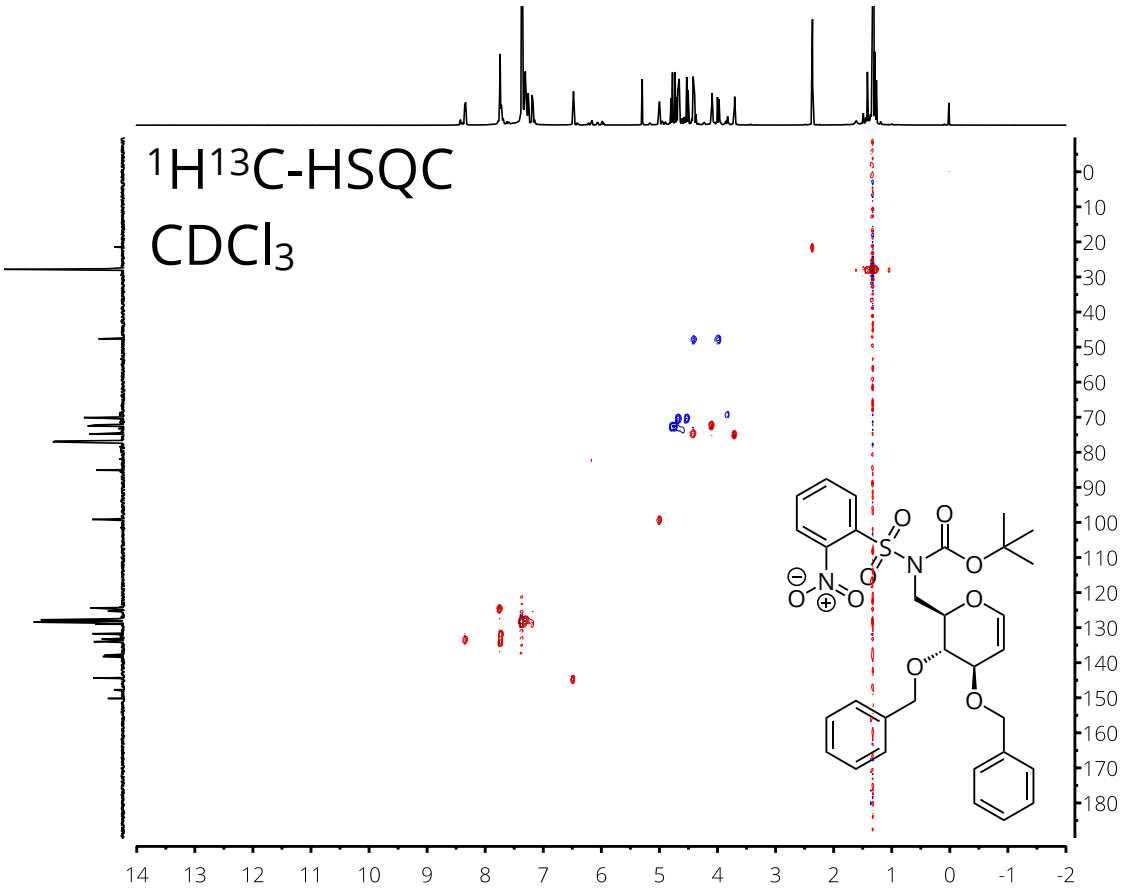
Trachea (porcine), Esophagus (ovine), and duodenum (porcine) were collected from freshly sacrificed animals (Animal Biotech Industries and Animal Technologies Inc.), immediately kept at 4 ºC, and transported to the site of the experiment. Tissues were gently washed in 10 mM PBS pH 7.4 and cut to expose the undisturbed interior luminal surface of the tissue. In the case of the porcine duodenum, the interior surface was gently washed again to remove remaining debris. Strips of tissue (2 x 11 x 5 mm) were cut and fixed onto a hydrophobic surface (e.g., Styrofoam) and imaged under long-wave UV illumination at a fixed (6 in) distance. The strips were then immersed in vials containing a 0.5 mg/mL solution of the fluorescently-labeled polymers for 30 minutes, gently washed with PBS, and fixed again onto the hydrophobic surfaces, followed by imaging. The surfaces with fixed tissue were washed 7 times by immersion in PBS buffer followed by vigorous shaking, with imaging under UV illumination between each wash. Images were collected and analyzed by ImageJ to extract the average intensity of red (TAMRA-**P3**) and green (FITC-dextran) in regions of interest for each tissue sample (n=3 for each polymer). These average intensities were compared against tissue incubated in PBS without fluorescent polymer (n=3) to generate fold increase in ROI intensity for each wash. For tissue loading experiments, strips of freshly cut tissue (trachea, esophagus, and duodenum) were immersed in a 0.5 mg/mL solution of TAMRA-**P3**, FITC-dextran, FITC-alginate, or FITC-dextran for 30 minutes. Then, the tissue samples were quickly dipped in PBS to remove excess dye/polymer solution and immersed in vials filled with 1 mL of 10 mM PBS pH 7.4. These vials were sonicated at high frequency for 1 hour in order to remove all bound polymer. After sonication, an aliquot (100 μL) of this solution for 3 samples was dispensed into a 96 well plate for measurement with a plate reader (λ_{ex} = 488 nm, λ_{em} = 520 nm and λ_{ex} = 541 nm, λ_{em} = 570 nm). Fluorescence readings were converted to mass loadings using a calibration curve obtained by serially diluting TAMRA-**P3**, FITC-dextran, FITC-alginate, or FITC-dextran in 10 mM PBS pH 7.4.

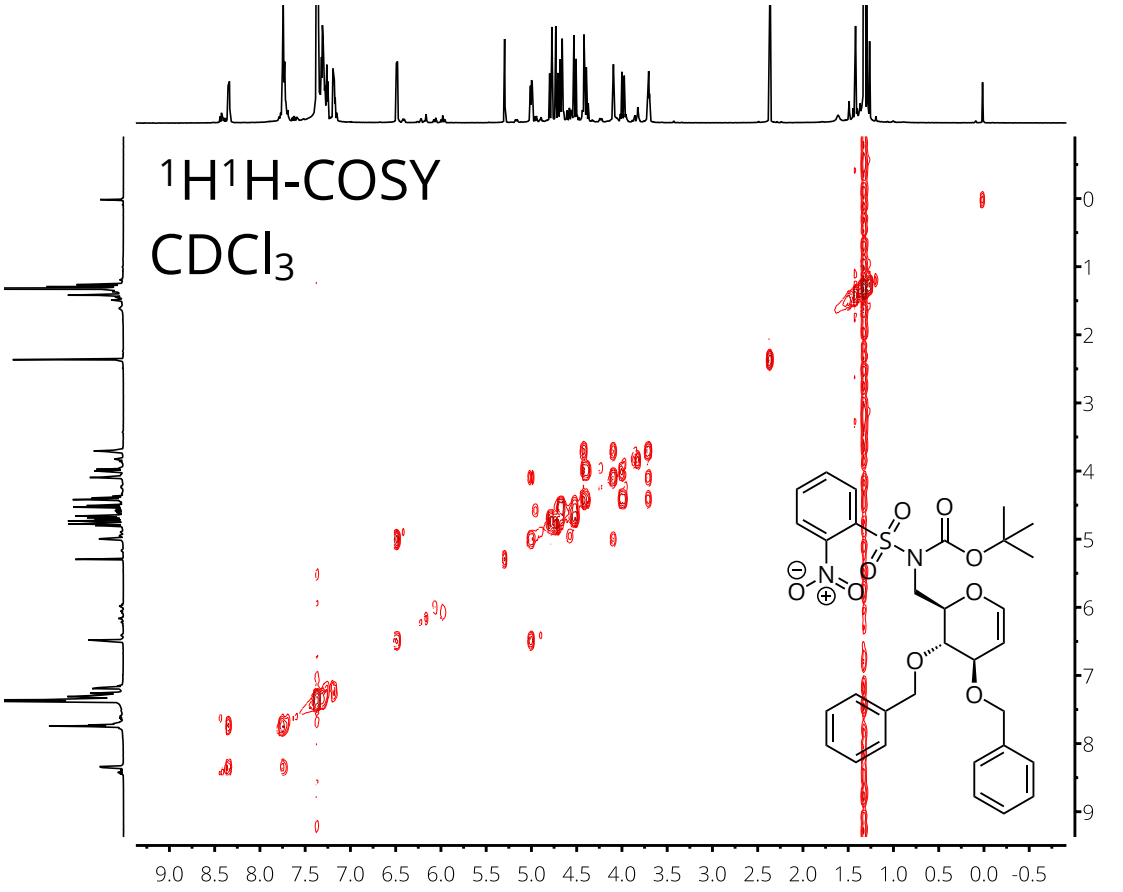
Spectroscopic characterization

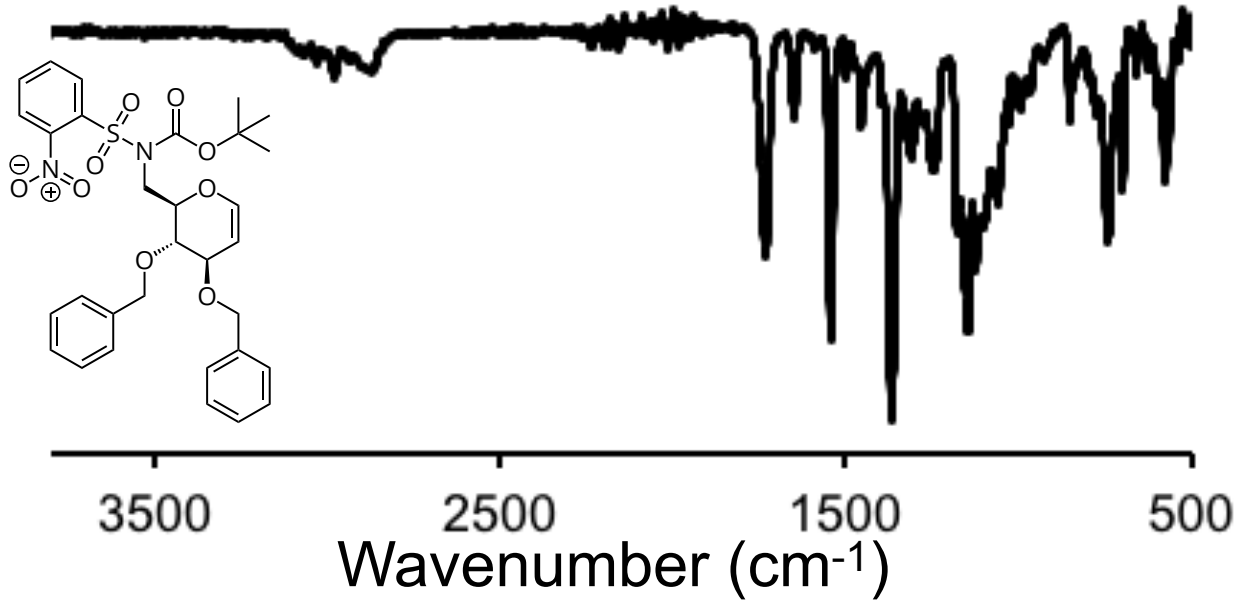
*(**5**) 3,4-di-O-benzyl-6-N-(2-nitrobenzenesulfonyl)-N-boc-6-deoxy-D-glucal.*





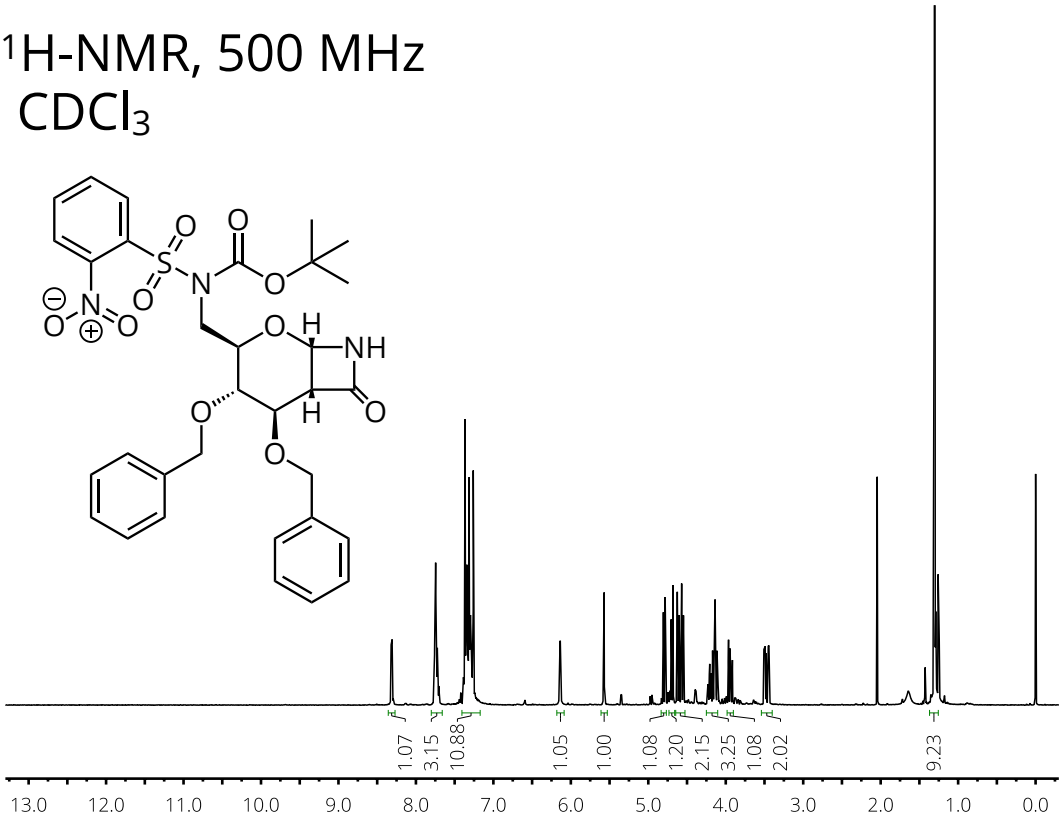

$^{1}H^{13}C$-HSQC
$CDCl_3$


$^{1}H^{1}H$-COSY
$CDCl_3$

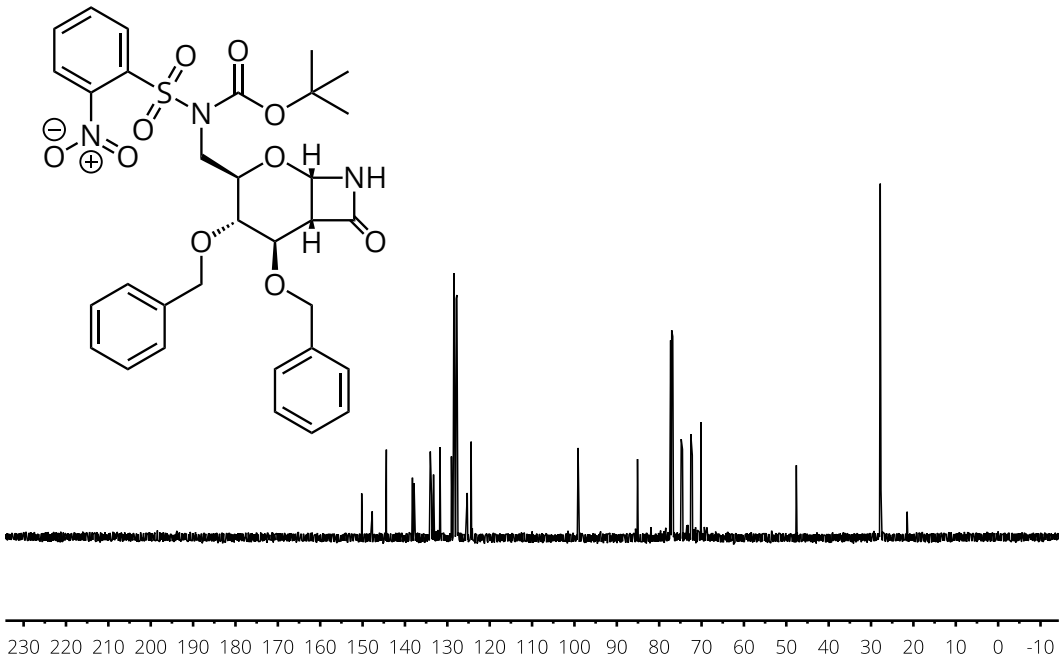

3500
2500
1500
500
Wavenumber (cm-1)

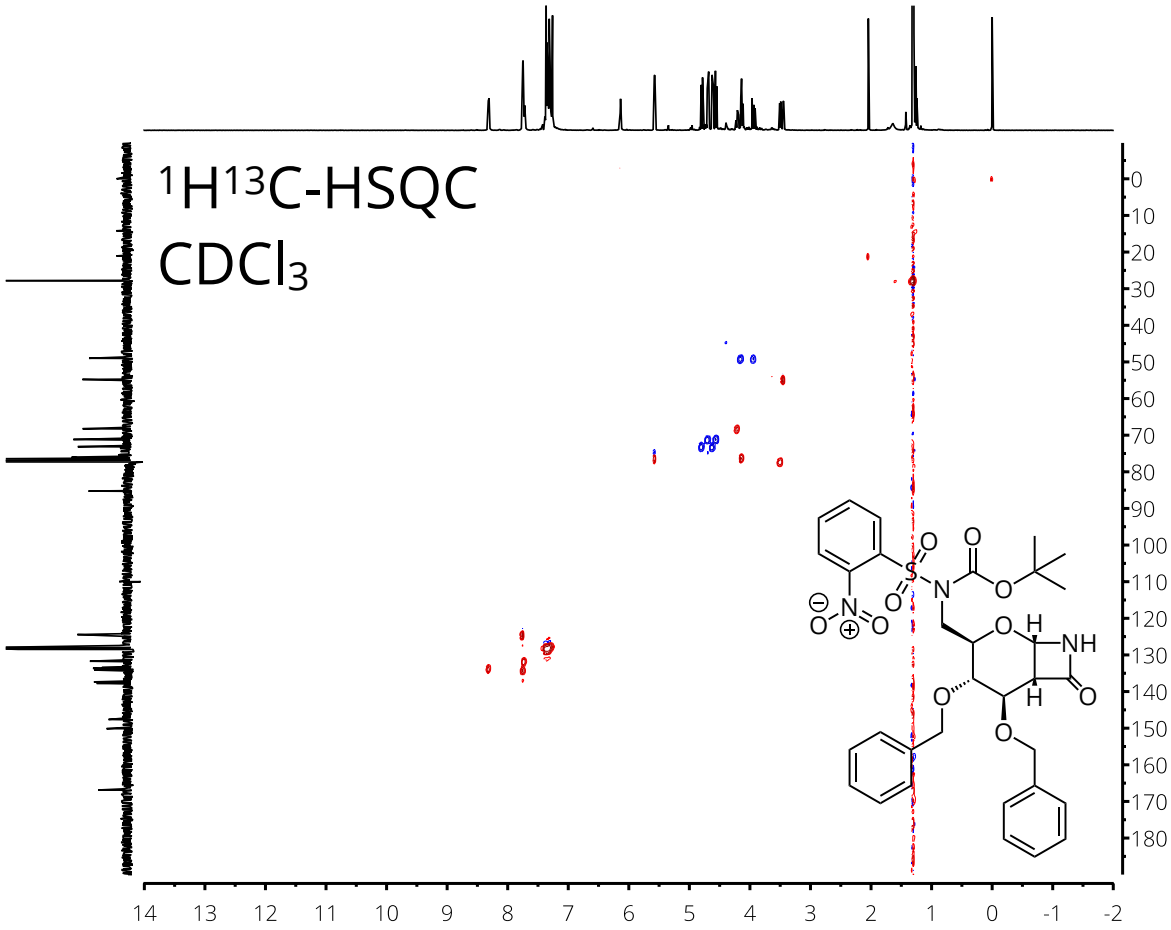
*(**6**) 1,2-β-lactam-3,4-di-O-benzyl-6-N-(2-nitrobenzenesulfonyl)-N-boc-6-deoxy-D-glucal*

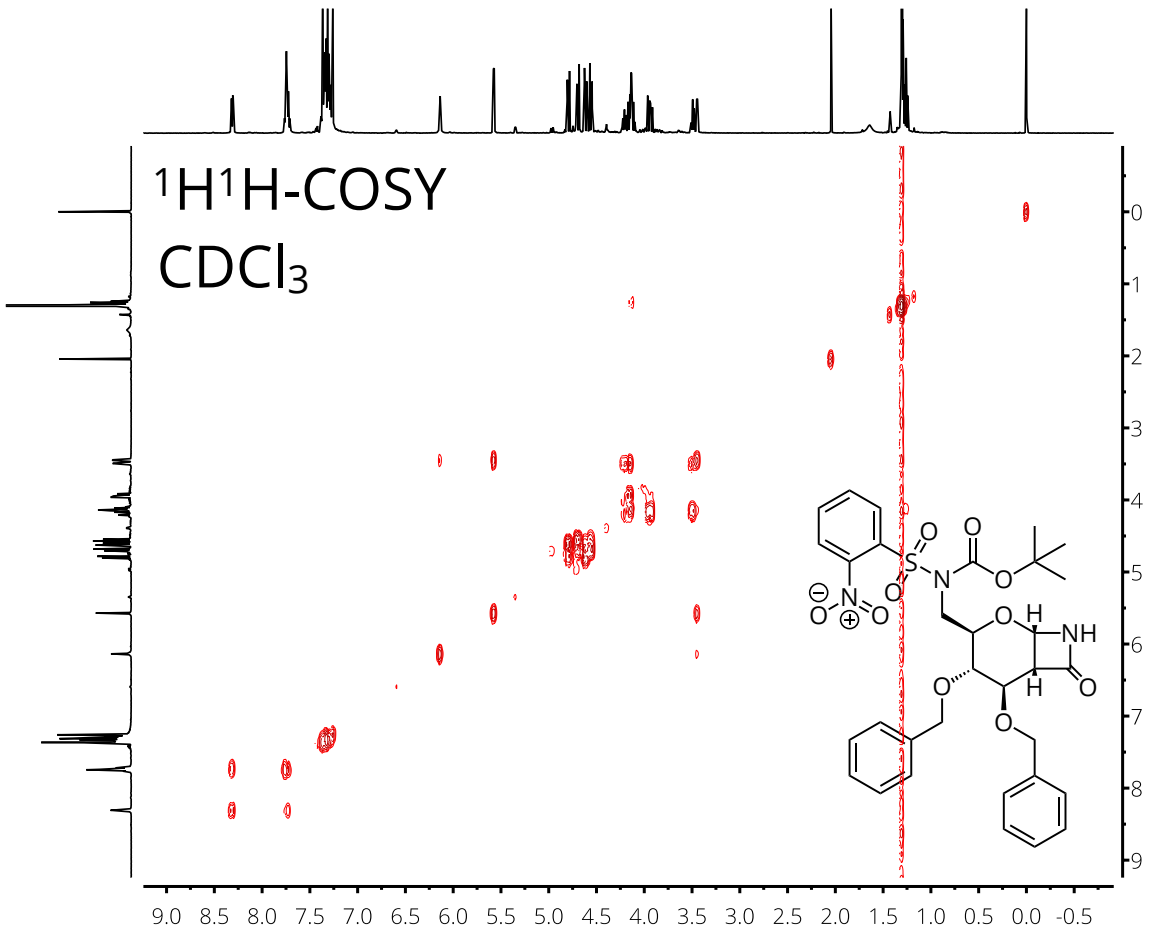
^{1}H-NMR, 500 MHz
$CDCl_3$

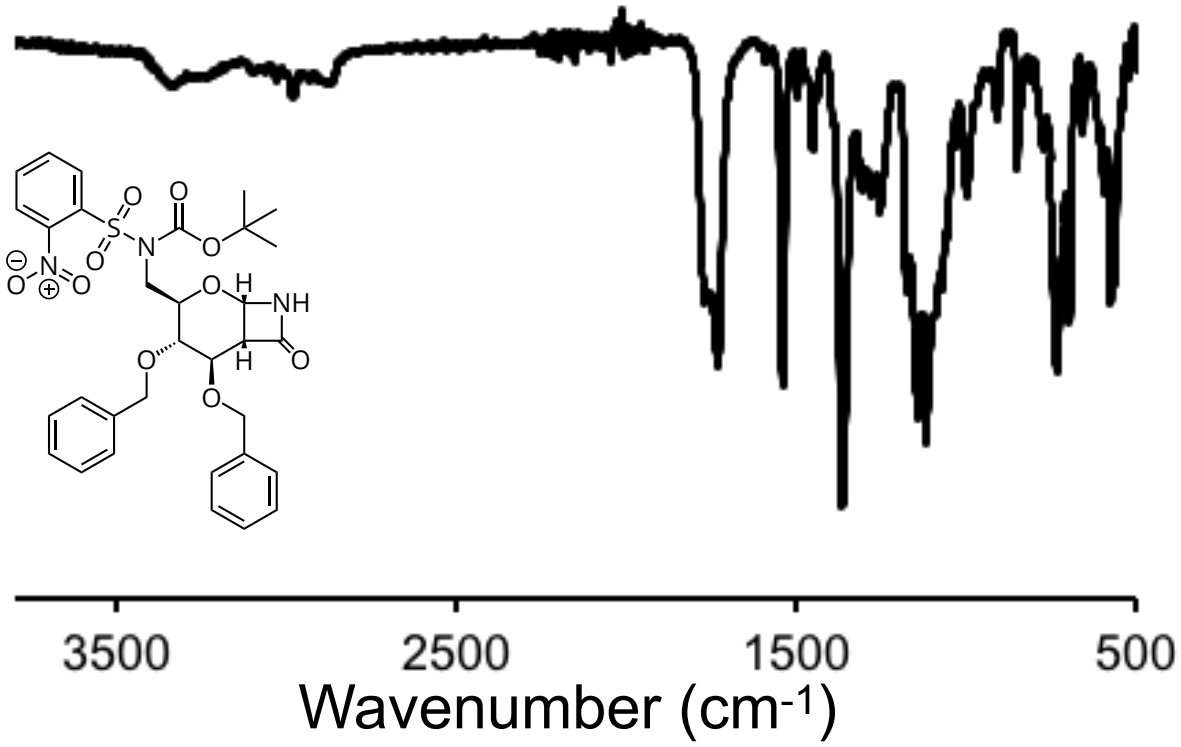


^{13}C-NMR, 125 MHz
$CDCl_3$

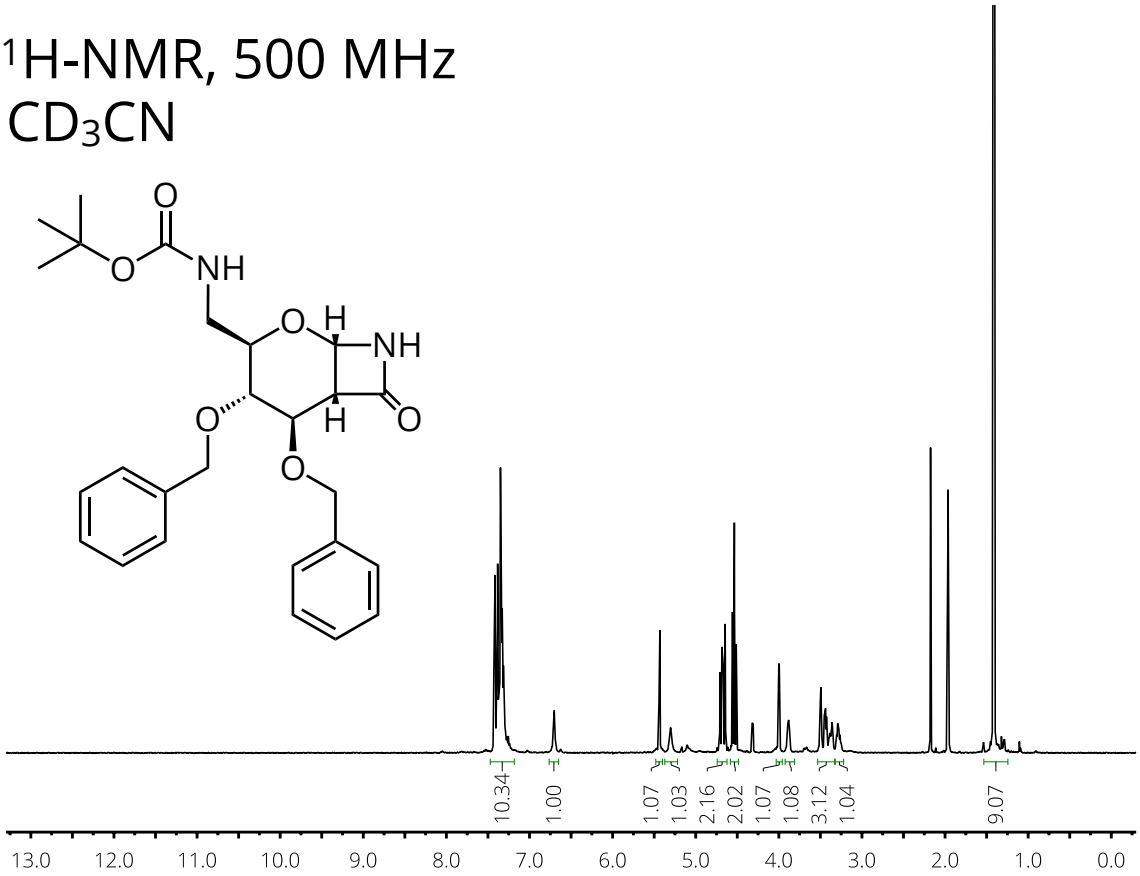


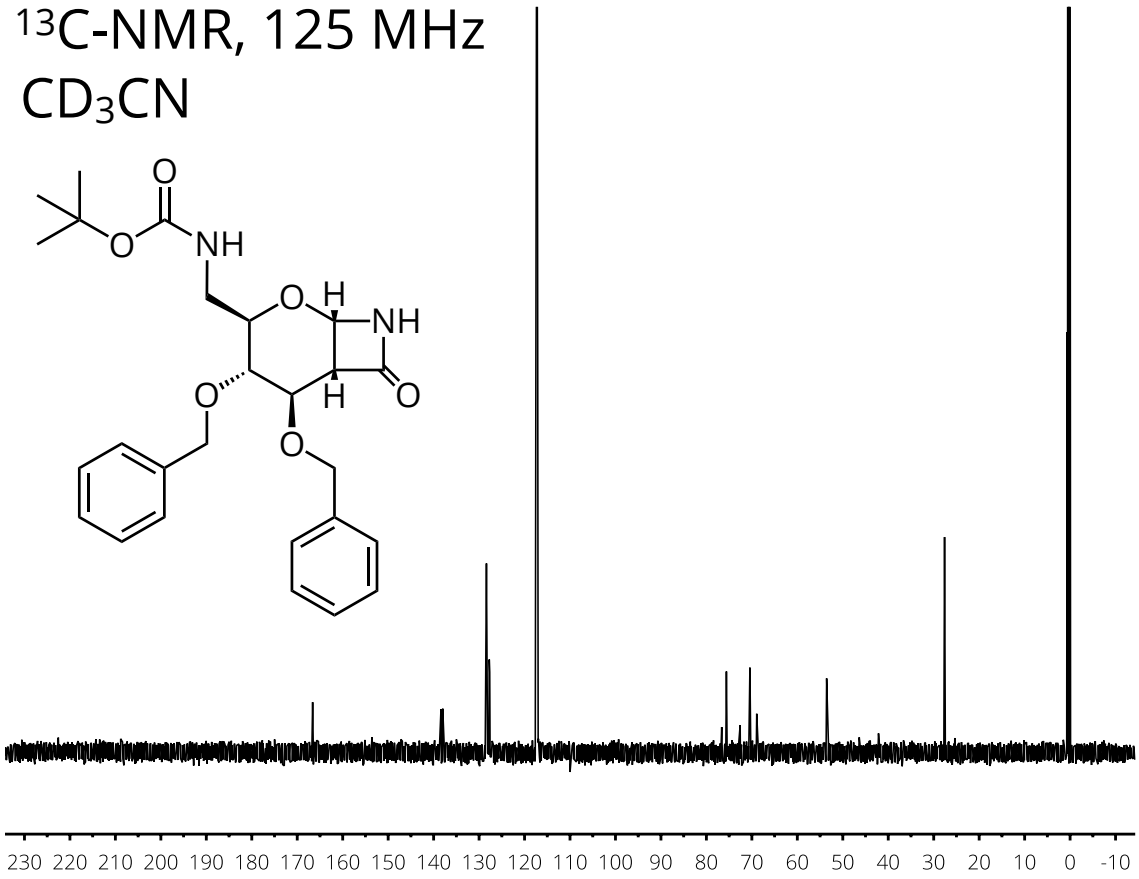

$^{1}H^{13}C$-HSQC
$CDCl_3$
14 13 12 11 10 9 8 7 6 5 4 3 2 1 0 -1 -2
0 10 20 30 40 50 60 70 80 90 100 110 120 130 140 150 160 170 180

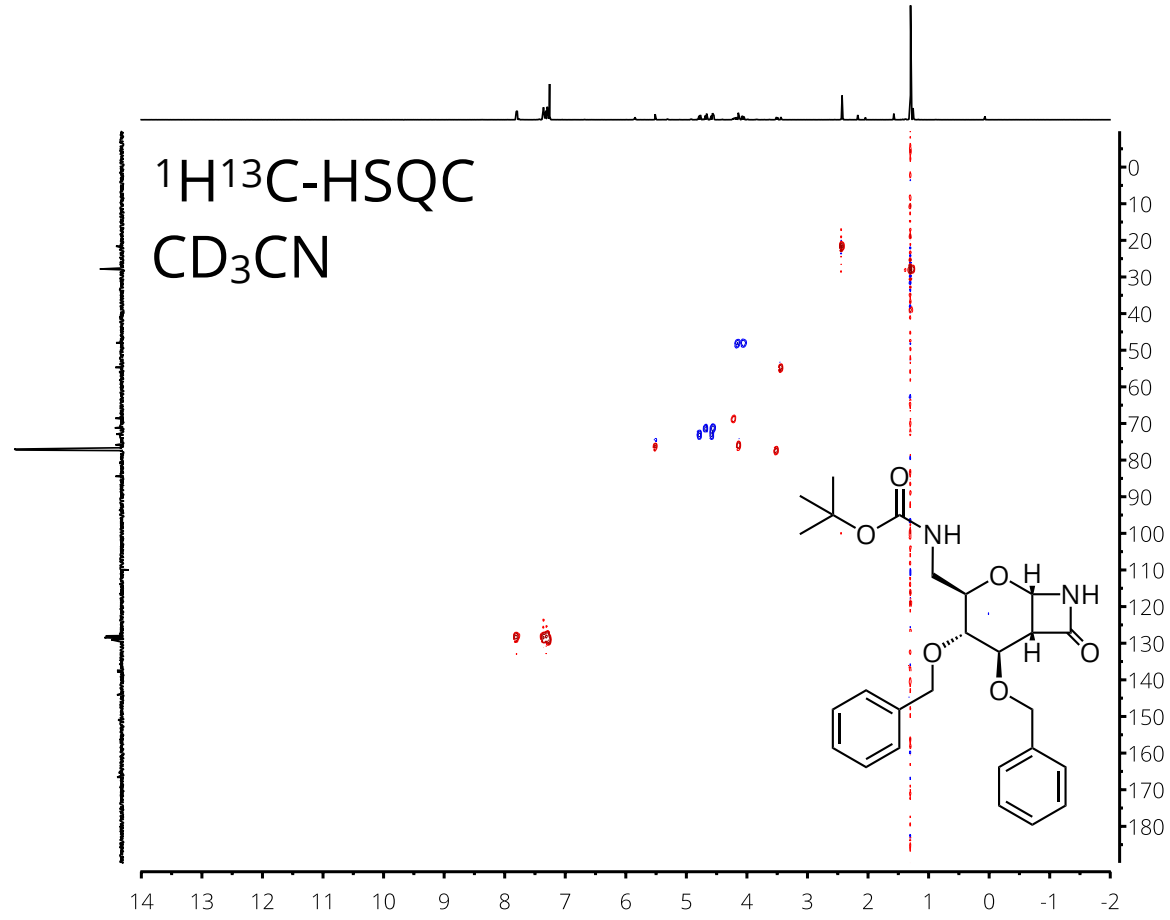

$^{1}H^{1}H$-COSY
$CDCl_3$
9.0 8.5 8.0 7.5 7.0 6.5 6.0 5.5 5.0 4.5 4.0 3.5 3.0 2.5 2.0 1.5 1.0 0.5 0.0 -0.5
0 1 2 3 4 5 6 7 8 9

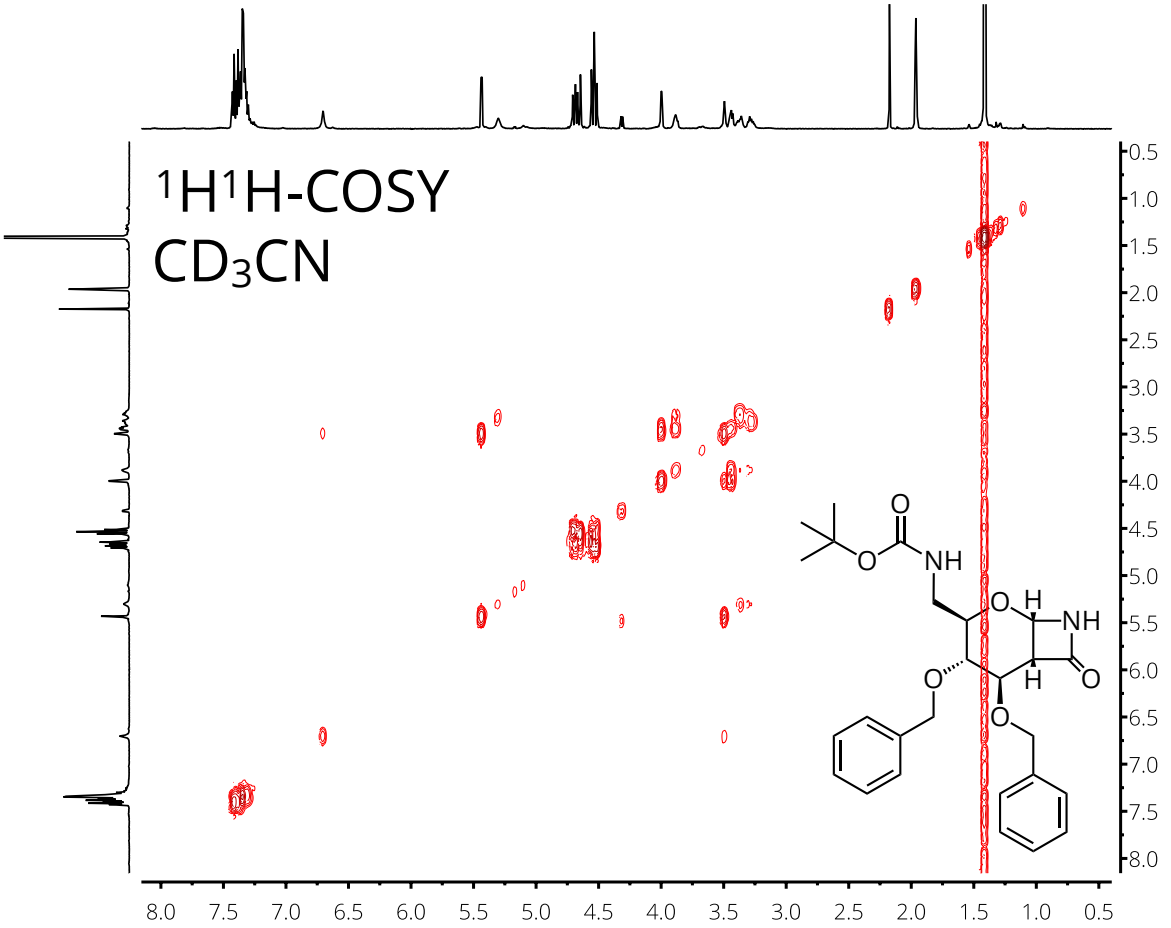

3500
2500
1500
500
Wavenumber (cm-1)

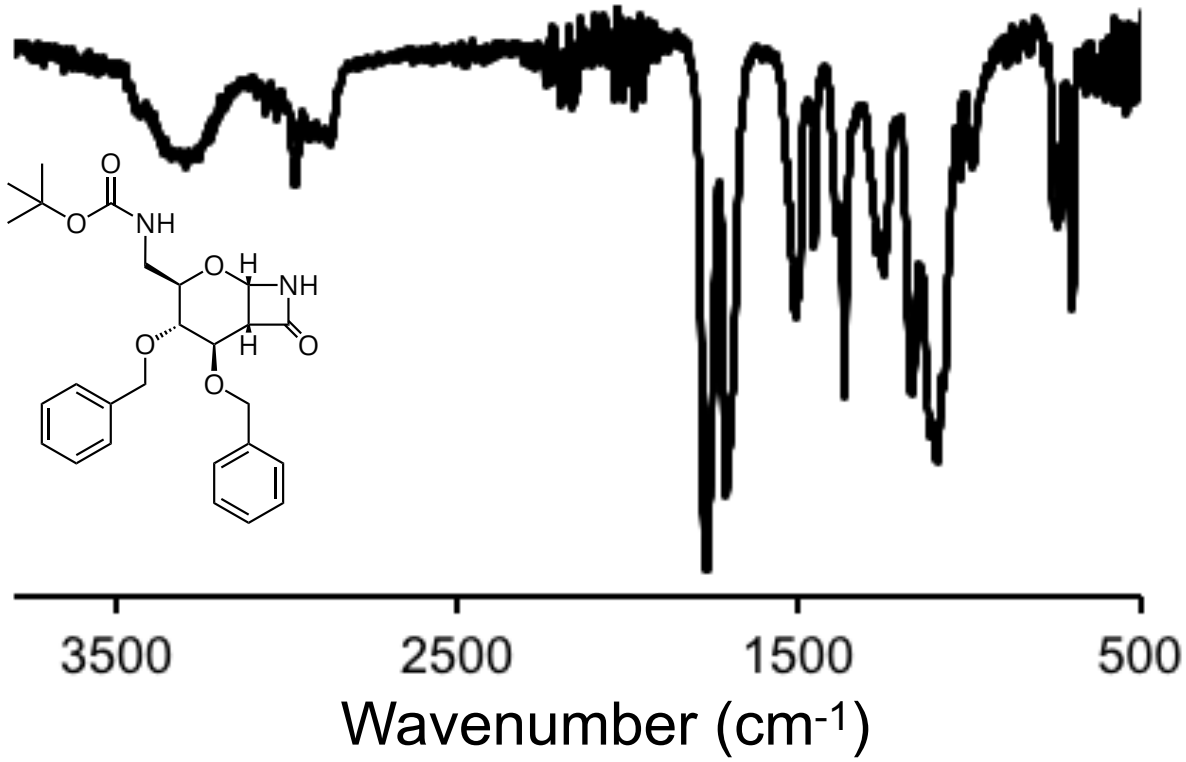
*(**7**) 1,2-β-lactam-3,4-di-O-benzyl-6-N-boc-6-deoxy-D-glucal.*





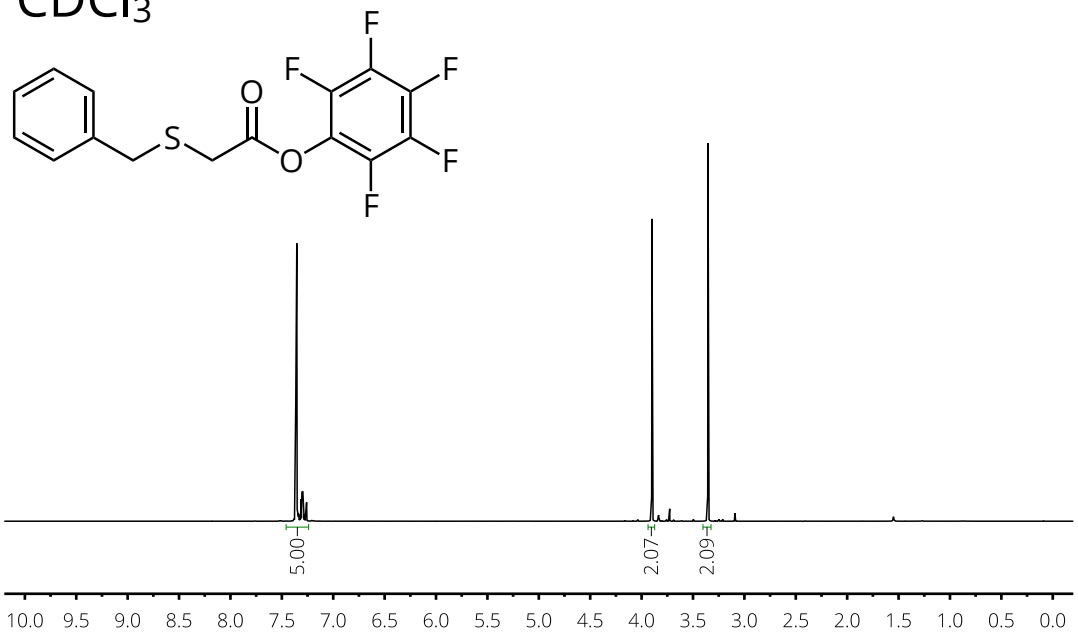

$^{1}H^{13}C$-HSQC
CD_3CN
O
O
NH
O
H
NH
O
H
O
O
O
0
10
20
30
40
50
60
70
80
90
100
110
120
130
140
150
160
170
180
14
13
12
11
10
9
8
7
6
5
4
3
2
1
0
-1
-2


$^{1}H^{1}H$-COSY
CD_3CN
O
O
NH
O
H
NH
O
H
O
O
O
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0
8.0
7.5
7.0
6.5
6.0
5.5
5.0
4.5
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5

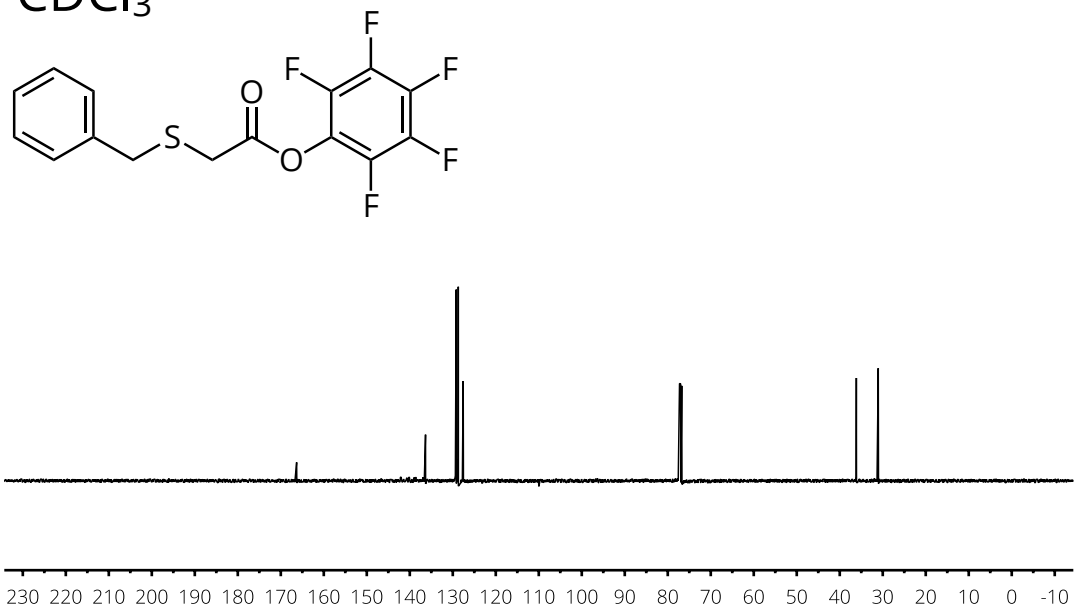

O
O
NH
O
H
NH
O
H
O
O
3500
2500
1500
500
Wavenumber (cm-1)

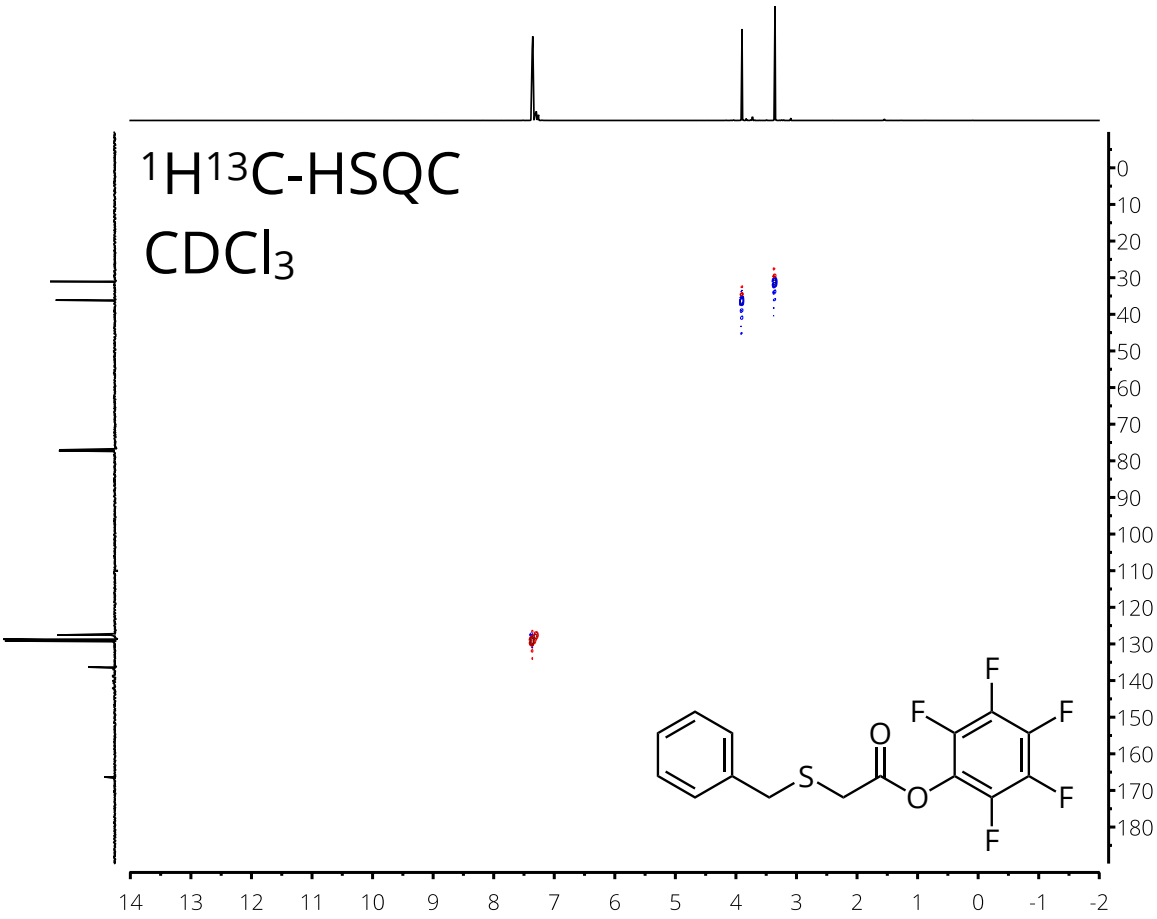
*(**9**) Pentafluorophenyl S-benzylthioglycolate.*

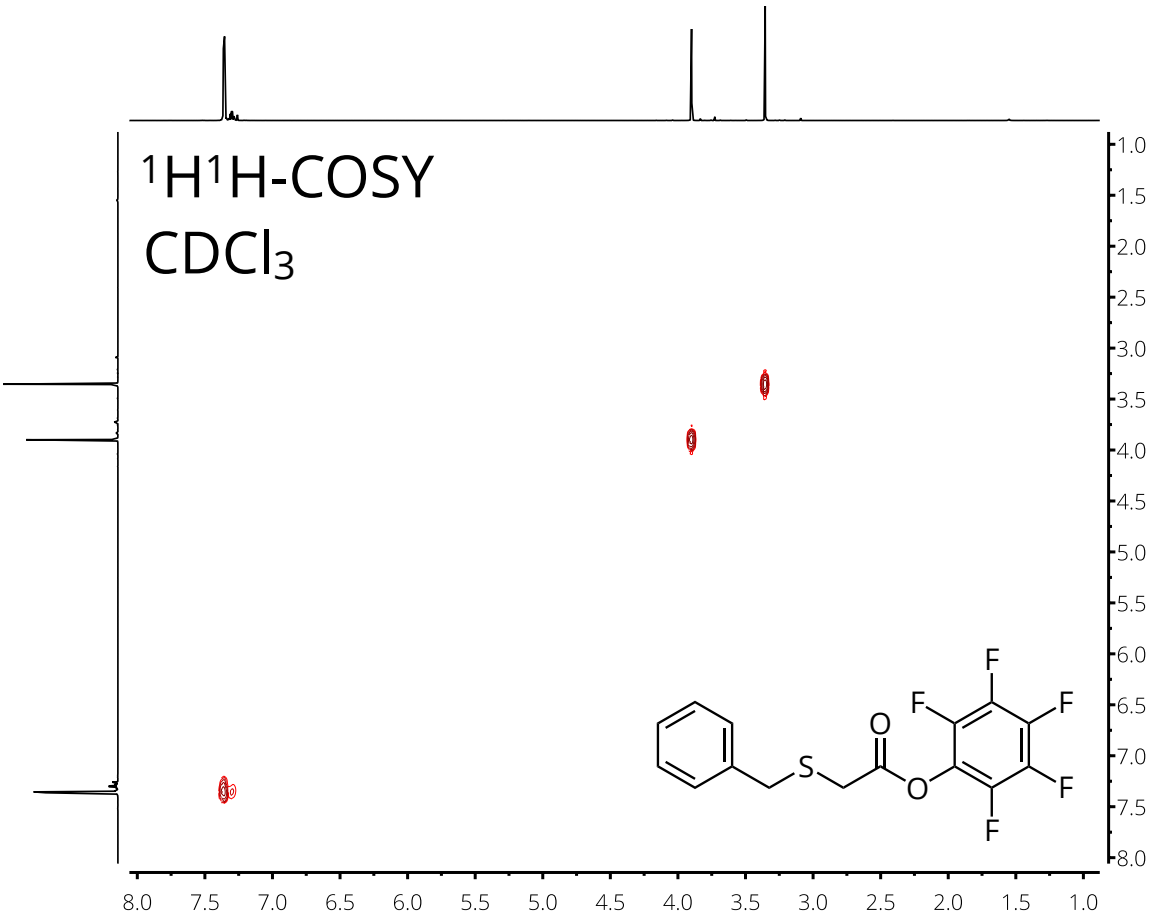
^{1}H-NMR, 500 MHz
$CDCl_3$

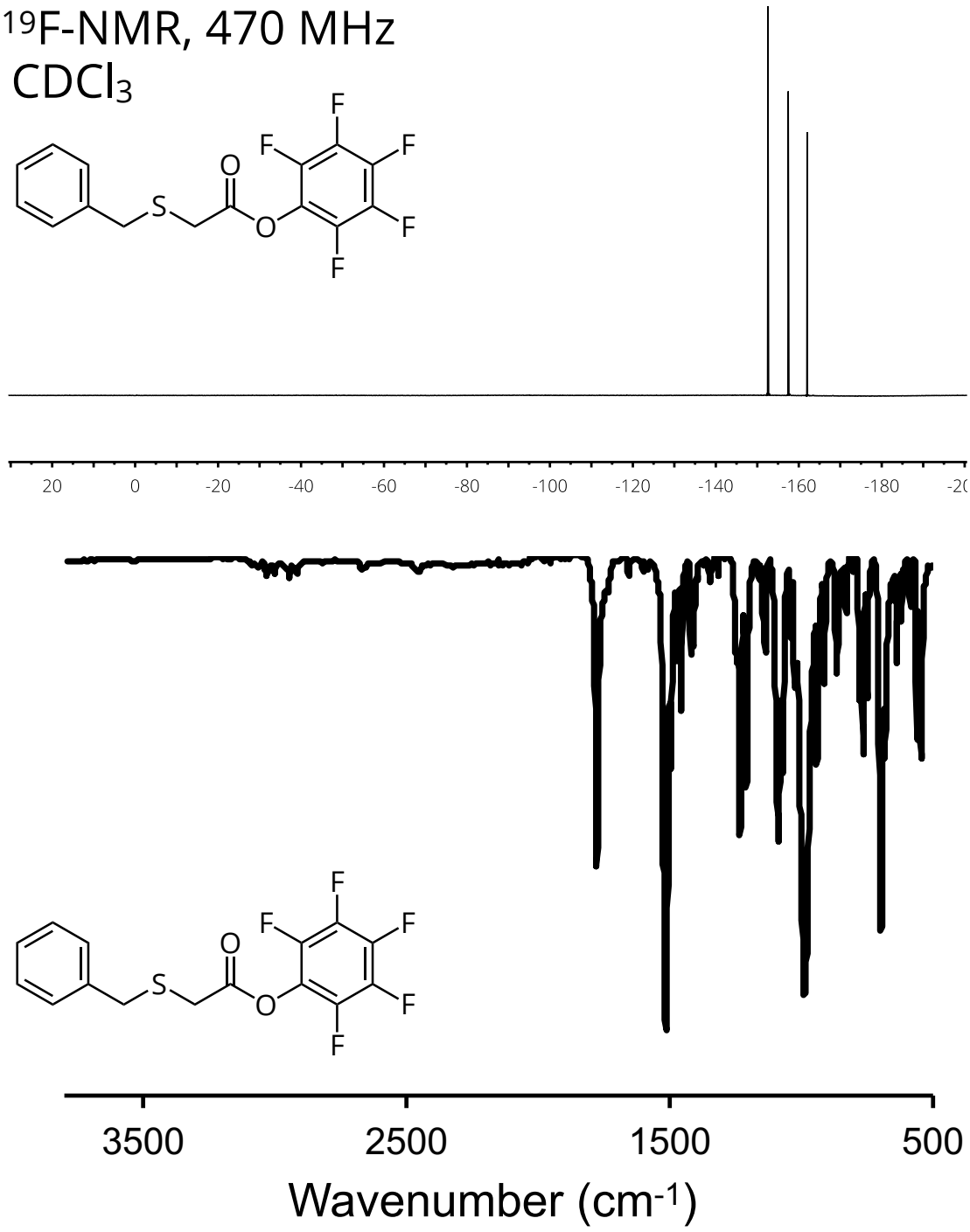


^{13}C-NMR, 125 MHz
$CDCl_3$

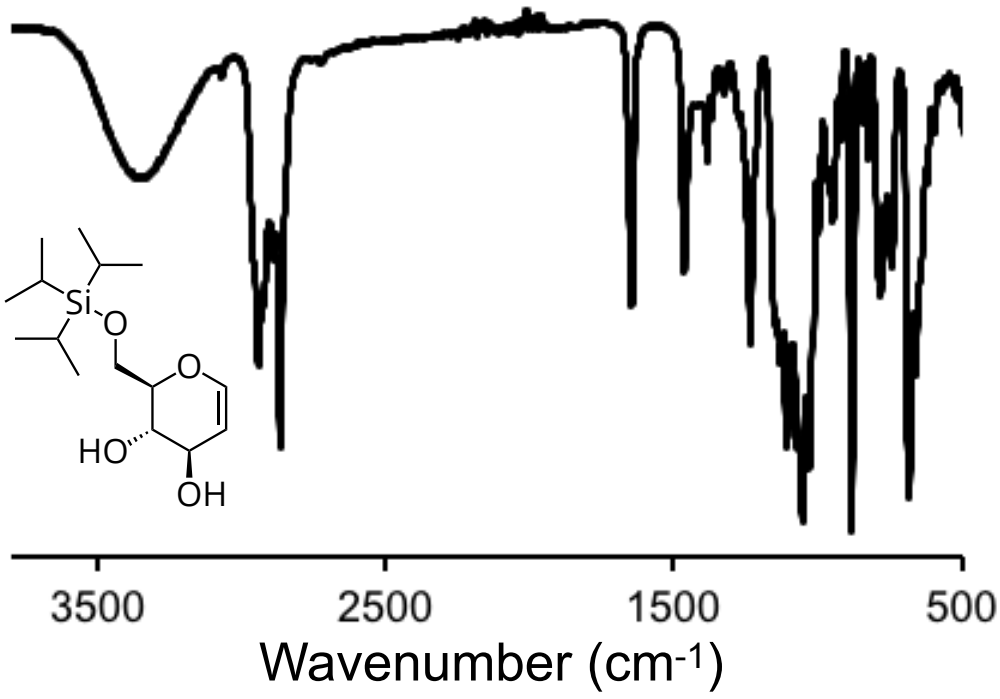


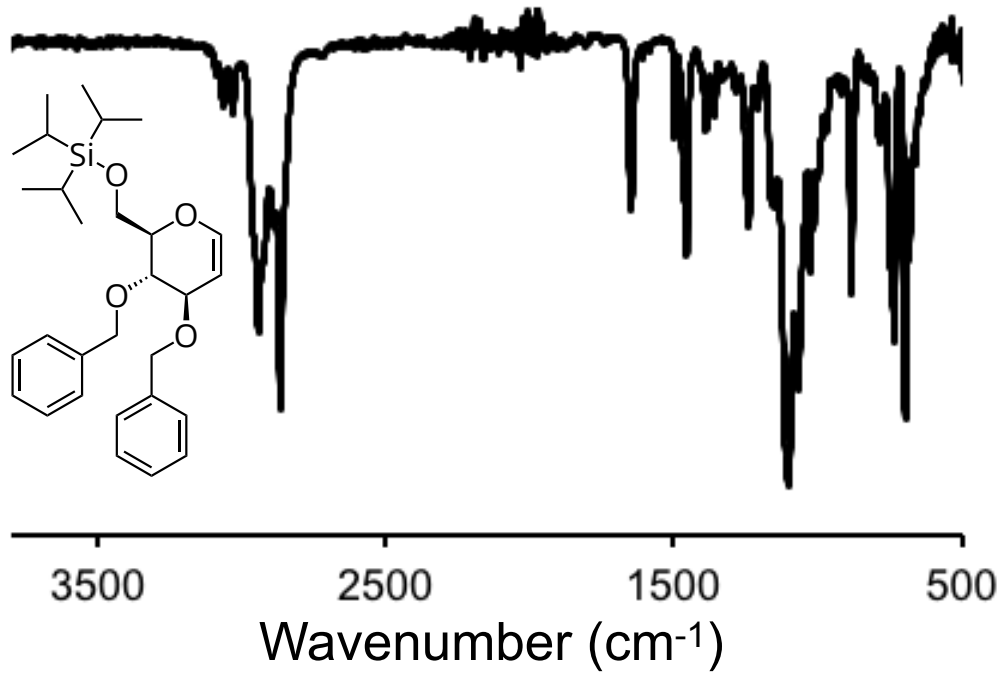

$^1H^{13}C$-HSQC
$CDCl_3$
F
F
F
F
F
O
O
S
14 13 12 11 10 9 8 7 6 5 4 3 2 1 0 -1 -2
0 10 20 30 40 50 60 70 80 90 100 110 120 130 140 150 160 170 180

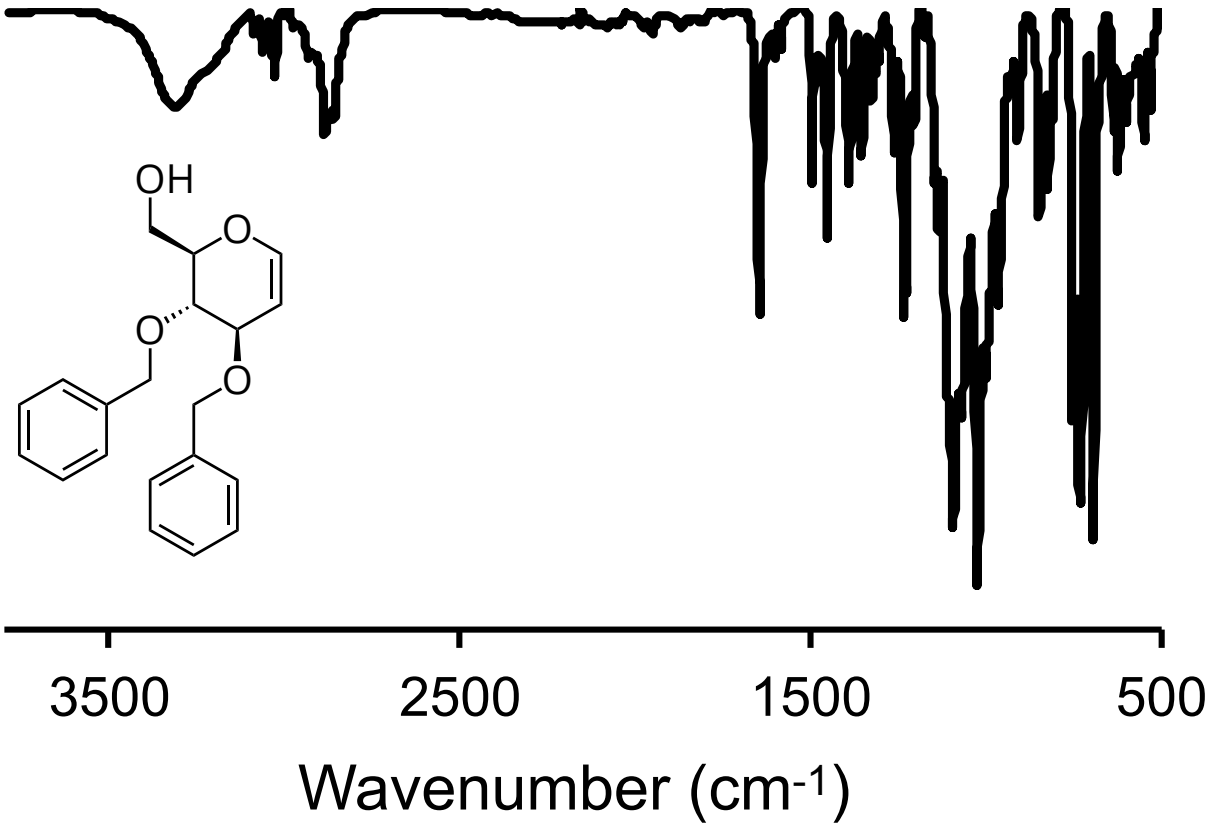

$^1H^1H$-COSY
$CDCl_3$
F
F
F
F
F
O
O
S
8.0 7.5 7.0 6.5 6.0 5.5 5.0 4.5 4.0 3.5 3.0 2.5 2.0 1.5 1.0
1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0 5.5 6.0 6.5 7.0 7.5 8.0


19F-NMR, 470 MHz
CDCl3
F
F
F
F
F
O
S
O
20
0
-20
-40
-60
-80
-100
-120
-140
-160
-180
3500
2500
1500
500
Wavenumber (cm-1)

IR Spectroscopic Data of Compounds Previously Described







MATLAB Code

BatchLoader – convert Excel data into MATLAB data structure and prune curves as needed.

```
%BatchLoader.m
disp('Beginning batch loading process...');
pause(2);
batchname = input('Input batch identifier to be analyzed: ','s');
NumBare = input('Input number of bare repetitions (equal to amine): ');
NumMucin = input('Input number of mucin repetitions: ');
save('BatchNumbers.mat','NumBare','NumMucin','batchname');
a = 1;
for i=1:(NumBare)
    FileNames{a} = strcat(batchname,'_Bare',num2str(i),'.xlsx');
    FinalName{a} = strcat(batchname,'_Bare',num2str(i),'_Raw.mat');
    a = a+1;
end
for i=1:(NumBare)
    FileNames{a} = strcat(batchname,'_Amine',num2str(i),'.xlsx');
    FinalName{a} = strcat(batchname,'_Amine',num2str(i),'_Raw.mat');
    a = a+1;
end
for i=1:(NumMucin)
    FileNames{a} = strcat(batchname,'_Mucin',num2str(i),'.xlsx');
    FinalName{a} = strcat(batchname,'_Mucin',num2str(i),'_Raw.mat');
    a = a+1;
end
a = a-1;
for i=1:a
```

```
    if ~exist(FileNames{i},'file')
        disp('One or more of the desired files could not be found... exiting
program.');
        return
    end
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

if exist('DatabaseInfoSav.mat','file')
    load('DatabaseInfoSav.mat');
    if batchname==batchinprogress
        cont = input('Found incomplete loading of the requested batch... continue
loading or restart? Y(1) R(2) Q(3): ');
        if cont==1
            swit = 2;
        elseif cont==2
            swit = 1;
            clear('DatabaseInfoSav.mat');
        elseif cont==3
            return
        end
    else
        disp('Saved running database does not match requested batch name...');
        cont = input('Delete running save files and begin new batch (1) or exit
program (2)?: ');
        if cont==1
            swit = 1;
            clear('DatabaseInfoSav.mat');
        elseif cont==2
            return
        end
    end
else
    swit = 4;
    for i=1:a
        if exist(FinalName{i},'file')&&swit==4
            fprintf('Loaded data already exists for %s ...\n',FinalName{i});
            cont = input('Continue to pruning (1), delete all data and restart (2),
or quit (3)?: ');
            if cont==1
                swit = 3;
                q = a+1;
            elseif cont==2
                swit = 1;
                clear('DatabaseInfoSav.mat');
            else
                return
            end
        end
    end
end
if swit==4
    swit =1;
end
if swit==1
    pause(1);
    q = 1;
    disp('Beginning data extraction from files...');
end
pause(1);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
batchinprogress = batchname;
while q<=a
    datafile = FileNames{q};
```

```
    fprintf('Reading from datafile %s ...\n',FileNames{q});
    disp('Reading header line...');
    NumRepsT = readtable(datafile,'Range','1:1','ReadVariableNames',false);
    NumReps = NumRepsT{1,1:end};
    SampleCount = max(NumReps);
    fprintf('The total number of replicates in this file is: %i\n',SampleCount);
    pause(2);
    disp('');
    LookupTable = ExcLookup(SampleCount);
    ToSave = strcat(FileNames{q}(1:end-5),'_Raw.mat');

save('DatabaseInfoSav.mat','datafile','SampleCount','LookupTable','q','FileNames','
batchinprogress','a');

    if swit==1
        disp('Beginning data extraction...');
        w = 1;
        u = 1;
        Ext = {};
        Ret = {};
        swit = 2;
    else
        disp('Resuming data extraction...');
        if exist(ToSave,'file')
            load(ToSave);
        else
            w = 1;
            u = 1;
        end
        Ext = {};
        Ret = {};
        swit = 2;
    end

    while (w<=SampleCount)
        fprintf('Extracting data from curve: %i\n(Press Cmd + . to quit)\n',w);
        pause(1);
        ExtRetFull =
readtable(datafile,'Range',LookupTable{w},'ReadVariableNames',false);
        Zext = cellfun(@str2num,ExtRetFull{3:end,1},'un',0);
        Fext = cellfun(@str2num,ExtRetFull{3:end,2},'un',0);
        Zret = cellfun(@str2num,ExtRetFull{3:end,3},'un',0);
        Fret = cellfun(@str2num,ExtRetFull{3:end,4},'un',0);
        Zext = Zext(~cellfun('isempty',Zext));
        Fext = Fext(~cellfun('isempty',Fext));
        Zret = Zret(~cellfun('isempty',Zret));
        Fret = Fret(~cellfun('isempty',Fret));
        Zext = cell2mat(Zext);
        Fext = cell2mat(Fext);
        Zret = cell2mat(Zret);
        Fret = cell2mat(Fret);
        Zext(Zext==0)=[];
        Fext(Fext==0)=[];
        Zret(Zret==0)=[];
        Fret(Fret==0)=[];
        disp('Data successfully loaded...');
        pause(1);
        Ext{1,1,u}(:)=Zext;
        Ext{1,2,u}(:)=Fext;
        Ret{1,1,u}(:)=Zret;
        Ret{1,2,u}(:)=Fret;
        u = u+1;
        w = w+1;
        save(ToSave,'u','w','Ext','Ret');
```

```
        fprintf('Saved force curve %i\n', w-1);
    end
    swit = 1;
    w = 1;
    u = 1;
    q = q+1;
end

if q>a
    if exist('DatabaseInfoSav.mat','file')
        WrkFile = fullfile(cd,'DatabaseInfoSav.mat');
        delete(WrkFile);
    end
    clear
    swit = 3;
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

disp('Beginning pruning process... exiting in the middle requires restarting
program');
load('BatchNumbers.mat');
pause(2);

a = 1;
for i=1:(NumBare)
    PrunedNames{a} = strcat(batchname,'_Bare',num2str(i),'_Pruned.mat');
    FinalName{a} = strcat(batchname,'_Bare',num2str(i),'_Raw.mat');
    a = a+1;
end
for i=1:(NumBare)
    PrunedNames{a} = strcat(batchname,'_Amine',num2str(i),'_Pruned.mat');
    FinalName{a} = strcat(batchname,'_Amine',num2str(i),'_Raw.mat');
    a = a+1;
end
for i=1:(NumMucin)
    PrunedNames{a} = strcat(batchname,'_Mucin',num2str(i),'_Pruned.mat');
    FinalName{a} = strcat(batchname,'_Mucin',num2str(i),'_Raw.mat');
    a = a+1;
end
a = a-1;
q = 1;
while q<=a
    load(FinalName{q});
    fprintf('Now showing curves from file %s ...\n',FinalName{q});
    PruRet = {};
    PruExt = {};
    u = 1;
    k = 1;
    while k<size(Ext,3)
        if
size(Ext{:,1,k},2)==size(Ext{:,2,k},2)&&size(Ret{:,1,k},2)==size(Ret{:,2,k},2)
            fprintf('Now showing graph %i\n',k);
            hold all
            plot(Ext{:,1,k},Ext{:,2,k},'LineWidth',4);
            plot(Ret{:,1,k},Ret{:,2,k},'LineWidth',4);
            set(gca,'Xdir','reverse');
            xlabel('Z position (m)');
            ylabel('Force reading (N)');
            set(gca,'linewidth',4,'fontsize',18,'fontname','Open Sans Light');
            set(gca,'TickDir','out');
            FixFigure(gcf)
            keeppr = '4';
            while (str2num(keeppr)~=1)&&(str2num(keeppr)~=2)&&(str2num(keeppr)~=3)
```

```
                keeppr = input('Keep this data? Y(1) N(2) Q(3): ','s');
                if isempty(keeppr)
                    keeppr = '6';
                end
            end
            close(gcf);
            chk = 3;
            while (chk==3)
                if str2num(keeppr)==1
                    PruExt(:,1,u) = Ext(:,1,k);
                    PruExt(:,2,u) = Ext(:,2,k);
                    PruRet(:,1,u) = Ret(:,1,k);
                    PruRet(:,2,u) = Ret(:,2,k);
                    save(PrunedNames{q},'u','k','PruExt','PruRet');
                    fprintf('Saved force curve %i...\n',k);
                    u = u+1;
                    chk = 2;
                elseif str2num(keeppr)==2
                    save(PrunedNames{q},'u','k','PruExt','PruRet');
                    fprintf('Discarding force curve %i...\n',k);
                    chk = 4;
                elseif str2num(keeppr)==3
                    chk = 2;
                    return
                else
                    return
                end
            end
        end
        k = k+1;
    end
    q = q+1;
end
pause(1)
disp('Loading process is complete... transition to NewCurveAnalysis');
```

NewCurveAnalysis – Analyze pruned data for inflection point, identify peaks, and extract quantitative data.

```
%New Curve Analysis

clear
disp('Beginning curve analysis process...');
pause(2);
FileName = input('Input filename (..."_Pruned.mat") to be analyzed: ','s');
if ~exist(FileName,'file')
    disp('One or more of the desired files could not be found... exiting
program.');
    return
end
Filehead = FileName(1:end-11);
ToSave = strcat(Filehead,'_PrePeak.mat');
q = 1;
pause(1);
contin = 2;
save('tempsave.mat','FileName');
if exist(ToSave,'file')
    disp('Scanned and processed data was found in directory...');
    contin = input('Continue to peak finding or restart? Y(1) N(2): ');
end
load(FileName);
A = size(PruExt);
i = 1;
disp('Scanning through force curves and finding inflection points...');
if contin==2
```

```
while i<=A(3)
    Offset = 0;
    InflIndex(i)=0;
    IndRetInfl(i)=0;
    InflXRet(i)=0;
    SampleDepth(i)=0;
    dF = [];
    if size(PruExt{:,1,i})==size(PruExt{:,2,i})
        DiffWindow = 10;
        Der = [];
        dY = 0;
        dX = 0;
        for p=1:(size(PruExt{:,1,i},2)-DiffWindow)
            for q=1:DiffWindow-1
                dY = PruExt{:,2,i}(p+q)-PruExt{:,2,i}(p);
                dX = PruExt{:,1,i}(p+q)-PruExt{:,1,i}(p);
                Der(q)=dY/dX;
            end
            dF(p) = mean(Der);
        end
        MeanDer = mean(dF(1:50*DiffWindow));
        StdDer = std(dF(1:50*DiffWindow));
        MeanVal = mean(PruExt{:,2,i}(1:50*DiffWindow));
        StdVal = std(PruExt{:,2,i}(1:50*DiffWindow));
        count = 0;
        p = 1;
        stopsig = 0;
        while p<=length(dF)&&stopsig==0
            CurrdF = dF(p);
            CurrVal = mean(PruExt{:,2,i}(p:p+DiffWindow));
            if CurrdF>(MeanDer+2*StdDer)||CurrVal>(MeanVal+10*StdVal)
                count = count+1;
            else
                count = 0;
            end
            if count>=DiffWindow
                stopsig = 1;
            end
            p = p+1;
        end
        if stopsig==0
            fprintf('No inflection point found for sample %d ...\n',i);
        elseif stopsig==1
            InflIndex(i) = p;
            OffsetExt = PruExt{:,1,i}(InflIndex(i))-0;
            ExtInflXVal = PruExt{:,1,i}(InflIndex(i));
            ExtInflYVal = PruExt{:,2,i}(InflIndex(i));
            [T,InflXRet(i)] = min(abs(PruRet{:,1,i}-ExtInflXVal));
            OffsetRet = PruRet{:,1,i}(InflXRet(i))-0;
            IndRetInfl(i) = size(PruRet{:,1,i},2);
            PruExt{:,1,i} = -1*(PruExt{:,1,i})+OffsetExt;
            PruRet{:,1,i} = -1*(PruRet{:,1,i}-OffsetRet);
            SampleDepth(i) = -1*min(PruExt{:,1,i});
        end
    end
    i = i+1;
end
save(ToSave,'PruRet','PruExt','SampleDepth','i','IndRetInfl','InflIndex','InflXRet'
,'ToSave');
end
clear

load('tempsave.mat');
Filehead = FileName(1:end-11);
```

```
ToSave = strcat(Filehead,'_PrePeak.mat');
disp('Please input all curves that you would like to analyze for this data set:');
selprom = 0;
CurveNum = 0;
i = 1;
load(ToSave);
while selprom==0
    inp = input('Input curve number, d if done, a for all curves, and q to quit:
','s');
    if inp=='d'
        selprom = 1;
    elseif inp=='q'
        return
    elseif inp=='a'
        CurveNum = 1:length(IndRetInfl);
        selprom = 1;
    else
        CurveNum(i) = str2double(inp);
        i = i+1;
    end
end
pau = input('Wait for input between force curves? Y(1) N(2) Q(3): ');
if pau==3
    return
end

x = 1;
b = 1;
v = 1;
MasterMaxForce=[];
w = 1;
f = 0;
PeakRuptLen = [];
while w<=length(CurveNum)
%Load the curve, x denotes which curve it is
x = CurveNum(w);
%b and bstart are counters for curve-dependent variables
bstart = b;
if ((InflXRet(x)+500)<size(PruRet{:,1,x},2))&&InflXRet(x)>0
    %If there is at least 500 points from inflection to end and the
    %inflection point is not 0
    test = sum((PruRet{:,1,x}(1:end-1)-PruRet{:,1,x}(2:end))==0);
    if test~=0
        MaxXValue = PruRet{:,1,x}(end);
        MinXValue = PruRet{:,1,x}(1);
        NumPoints = size(PruRet{:,1,x},2);
        PruRet{:,1,x} =
linspace(PruRet{:,1,x}(1),PruRet{:,1,x}(end),size(PruRet{:,1,x},2));
        chk=1;
    else
        chk=1;
    end
    if chk==1
    if (size(PruRet{:,1,x})==size(PruRet{:,2,x}))
        %Subtract Retraction Curve from Extension Curve
        ReversedExt = flip(PruExt{:,2,x});
        UpperLength = min([size(ReversedExt,2) size(PruRet{:,2,x},2)]);
        ReversedExt = ReversedExt(1:UpperLength);
        %The Extension curve is usually shorter - set this as upper limit
        ExaminedRet = PruRet{:,2,x}(1:UpperLength);
        ExaminedRetX = PruRet{:,1,x}(1:UpperLength);
        Subtracted = ReversedExt - ExaminedRet;
        %Set the inflection point as the lower limit
        Subtracted = Subtracted(InflXRet(x):end);
```

```
ExaminedRetX = ExaminedRetX(InflXRet(x):end);
%Determine the mean and stdev of the subtracted curves
b = 1;
p = length(Subtracted);
MeanSub = [];
StdSub = [];
while p>1
    if b>100
        if Subtracted(p)<=(MeanSub+5*StdSub)
            ToTest(b) = Subtracted(p);
            MeanSub = mean(ToTest);
            StdSub = std(ToTest);
            b = b+1;
        end
    else
        ToTest(b) = Subtracted(p);
        MeanSub = mean(ToTest);
        StdSub = std(ToTest);
        b = b+1;
    end
    p = p-1;
end
Threshold = MeanSub+2*StdSub;
%Hysteresis Curve: X = ExaminedRetX , Y = Subtracted
%Peaks are regions with sharply negative derivative - the width of
%the sliding window determines the granularity
%Calculate the window derivative at each point and mean and
%standard deviation of derivative for whole curve
DiffWindow = 10;
Der = [];
dY = 0;
dX = 0;
dF = [];
%Start from end of curve and work backwards to x=0 to remove
%outliers and get true mean and std of derivatives
p = length(Subtracted);
a = 1;
MeanDer = [];
while p>=(DiffWindow)
    for q=1:DiffWindow-1
        dY = Subtracted(p)-Subtracted(p-q);
        dX = ExaminedRetX(p)-ExaminedRetX(p-q);
        Der(q) = dY/dX;
    end
    if ~isempty(MeanDer)
        %Once there is a mean and stdev
        if mean(Der)>=(MeanDer-3.5*StdDer)
            %If the local derivative is not an outlier, add it
            TempMeanDer(a) = mean(Der);
            a = a+1;
        end
    else
        %At the beginning, add all local derivatives to the total
        TempMeanDer(a) = mean(Der);
        a = a+1;
    end
    if a>=100
        %Once enough derivatives have been sampled
        MeanDer = mean(TempMeanDer);
        StdDer = std(TempMeanDer);
    end
    p = p-1;
end
```

```
        for p=1:(length(Subtracted)-DiffWindow)
            %Go back through in the forward direction for indexing purposes
            for q=1:DiffWindow-1
                dY = Subtracted(p+q)-Subtracted(p);
                dX = ExaminedRetX(p+q)-ExaminedRetX(p);
                Der(q)=dY/dX;
            end
            dF(p) = mean(Der);
        end
        %Points where the derivative is less than mean-3.5*stdev are near
        %peaks
        g = 1;
        for p=DiffWindow:(length(Subtracted)-DiffWindow)
            if dF(p)<(MeanDer-3.5*StdDer)
                Peaks{g} = [ExaminedRetX(p) Subtracted(p) p];
                g = g+1;
            end
        end

        %Each point of negative derivative should correspond to one
        %previous local maximum (the peak). Points within one DiffWindow
        %in either direction should correspond to distinct peaks.
        %Therefore, search for maximum within one DiffWindow on either side
        %of the hit point - since each peak has a unique X value, only 1
        %hit point can correspond to that unique X value.
        NewPeaks = [];
        v = 1;
        for a=1:g-1
            %Initial guess of peak
            Center = Peaks{a}(3);
            Ind = 0;
            while Ind~=(DiffWindow+1)&&(Center-
DiffWindow)>=1&&(Center+DiffWindow)<=length(Subtracted)
                %Window to look for maximum
                low = Center-DiffWindow;
                hi = Center+DiffWindow;
                %Find Maximum Y value of the window
                [Max,Ind] = max(Subtracted(low:hi));
                Center = low+Ind-1;
            end
            if Center-DiffWindow<1
                low = 1;
                hi = DiffWindow;
                [Max,Ind] = max(Subtracted(low:hi));
                Center = low+Ind-1;
            end
            if ~isempty(NewPeaks)
                if ~ismember(ExaminedRetX(Center),NewPeaks(:,1))
                    ChkWidth = sum(Subtracted(low:hi)>Threshold);
                    if (ChkWidth>=7)&&Center<(length(Subtracted)-5*DiffWindow)
                        %Peaks have to be unique and wide enough to be
                        %added
                        NewPeaks(v,:) = [ExaminedRetX(Center) Subtracted(Center)
Center];
                        v = v+1;
                    end
                end
            else
                ChkWidth = sum(Subtracted(low:hi)>Threshold);
                if (ChkWidth>=7)&&Center<(length(Subtracted)-5*DiffWindow)
                    NewPeaks(v,:) = [ExaminedRetX(Center) Subtracted(Center)
Center];
                    v = v+1;
                end
```

```
            end
        end

        %The coordinates of all peaks have been found and index is noted in
        %variable "Center"
        %Now have to find the left and right bounds of the peak.
        %For the right/upper bound - values should be below threshold for
        %some time before the peak is considered ended. Otherwise, if
        %another peak is encountered before the current peak upper bound
        %has been determined, it will be split into a subpeak
        spi = 1;
        i = 1;
        PeakNum = 0;
        Bounds = [1];
        PeakDividers = {};
        if ~isempty(NewPeaks)
            while i<=size(NewPeaks,1)
                a = NewPeaks(i,3); %a is the starting position / peak
                count = 0;
                MaxLimit = length(Subtracted)-3*DiffWindow;
                while a<=MaxLimit
                    a = a+1; %move one position to the right
                    if Subtracted(a)<Threshold %is the value below threshold
                        count = count +1; %If so, start the counter
                    else
                        %if it goes above the threshold, restart counter
                        count = 0;
                    end
                    if
i+1<=size(NewPeaks,1)&&(count<3*DiffWindow)&&ExaminedRetX(a)==NewPeaks(i+1,1) %If
the examined point hit another peak before first peak ended
                        spi = spi+1;
                        if spi==2
                            Bounds(spi) = NewPeaks(i,3);
                            spi = spi+1;
                        end
                        Bounds(spi) = a; %add the index to the list of bounds for
this peak
                        count = 0; %reset the counter
                        i = i+1; %skip to the next peak
                        a = length(Subtracted); %break from loop
                    elseif count>=3*DiffWindow %has the counter hit 3x the
DiffWindow
                        spi = spi+1;
                        Bounds(spi) = a; %If it has, stop the counting; a denotes
peak upper limit

                        %Perform same analysis but going backwards from the
                        %peak - only do it from the first peak in a series
                        %or if not part of a series, then the calculated
                        %center

                        if spi<=2
                            c = NewPeaks(i,3);
                        else
                            c = Bounds(2);
                        end
                        count2 = 0;
                        while c>1
                            c = c-1;
                            if Subtracted(c)<Threshold %is the value below
threshold
                                count2 = count2 + 1; %If so, start the counter
                            else
```

```
                            %if it goes above the threshold, restart counter
                            count2 = 0;
                            end
                            if (count2>3*DiffWindow)||c==1
                                Bounds(1) = c; %found the lower bound
                                c = 0; %exit the loop
                                count2 = 0;
                            end
                        end

                        Tmp = [];
                        spi = 1; %reset divider counter
                        a = length(Subtracted); %break from loop
                        j = 1;
                        PeakNum = PeakNum + 1;
                        while j+1<=length(Bounds)
                            Tmp(j,:) = [Bounds(j) Bounds(j+1)];
                            j = j+1;
                        end
                        Bounds = [1];
                        PeakDividers{PeakNum} = Tmp;
                        i = i+1;
                    end
                    if a==MaxLimit
                        i = i+1;
                    end
                end
            end
        end


        %The number of elements of PeakDividers is the number of discrete
        %peak events

        %The total number of rows of PeakDividers-1 is the number of total
        %peaks/subpeaks

% ------------------------------------------------------------------------%
% Analysis on the predefined curve regions

% Absolute maximum force for all peaks and subpeaks
PeakSlopes = {};
s = 0;
fprintf('Number of unique peaks: %i\t',PeakNum);
if PeakNum==0
    fprintf('\n');
end
for j=1:PeakNum
    PeakList = PeakDividers{j};
    if size(PeakList,1)==1
        Start = PeakList(1,1);
        End = PeakList(1,2);
        PeakArray = Subtracted(Start:End);
        PeakArrayX = ExaminedRetX(Start:End);
        dArea = (PeakArray-MeanSub).*PeakArrayX;
        f = f+1;
        [AbsPeakMax(f),maxind] = max(PeakArray);
        [AbsPeakMin(f),minind] = min(Subtracted(Start:Start+maxind-1));
        %Model elastic region linearly
        ExpY = Subtracted((Start+minind-1):(Start+maxind-1))';
        ExpX = zeros(length(ExpY),2);
        ExpX(:,1) = 1;
        ExpX(:,2) = ExaminedRetX((Start+minind-1):(Start+maxind-1))';
```

```
        ElasticSlope = ExpX\ExpY;
        SlopeLine = ExpX*ElasticSlope;
        s = s+1;
        PeakSlopes{s} = [ExpX(:,2) SlopeLine];

        RelPeakMax(f) = max(PeakArray)-MeanSub;
        PeakRuptLen(f) = ExaminedRetX(Start+maxind-1);
        PeakWidth(f) = ExaminedRetX(End)-ExaminedRetX(Start);
        PeakArea(f) = sum(dArea);
        SlopeCoeff(f) = ElasticSlope(2);
        CurveID(f) = w;
        MainPeak(f) = 1;
        fprintf('Unique peak %i\n', j);
    else
        fprintf('Unique peak %i, Number of subpeaks: %i\n', j, size(PeakList,1)-1);
        for k=2:size(PeakList,1)
            f = f+1;
            AbsPeakMax(f) = Subtracted(PeakList(k,1));
            RelPeakMax(f) = Subtracted(PeakList(k,1))-MeanSub;
            PeakRuptLen(f) = ExaminedRetX(PeakList(k,1));
            if k==size(PeakList,1)
                Start = PeakList(k-1,1);
                EndMin = PeakList(k-1,2);
                [AbsPeakMin(f),minind] = min(Subtracted(Start:EndMin));
                End = PeakList(k,2);
                ExpY = Subtracted(Start+minind-1:PeakList(k-1,2))';
                ExpX = zeros(length(ExpY),2);
                ExpX(:,1) = 1;
                ExpX(:,2) = ExaminedRetX(Start+minind-1:PeakList(k-1,2));
            else
                Start = PeakList(k-1,1);
                End = PeakList(k-1,2);
                [AbsPeakMin(f),minind] = min(Subtracted(Start:End));
                ExpY = Subtracted(Start+minind-1:End)';
                ExpX = zeros(length(ExpY),2);
                ExpX(:,1) = 1;
                ExpX(:,2) = ExaminedRetX(Start+minind-1:End);
            end
            ElasticSlope = ExpX\ExpY;
            SlopeLine = ExpX*ElasticSlope;
            s = s+1;
            PeakSlopes{s} = [ExpX(:,2) SlopeLine];
            PeakWidth(f) = ExaminedRetX(End)-ExaminedRetX(Start);
            PeakArray = Subtracted(Start:End);
            PeakArrayX = ExaminedRetX(Start:End);
            dArea = (PeakArray-MeanSub).*PeakArrayX;
            PeakArea(f) = sum(dArea);
            SlopeCoeff(f) = ElasticSlope(2);
            CurveID(f) = w;
            MainPeak(f) = 0;
        end
    end
end

% ------------------------------------------------------------------------%
        %plotting

        fig1 = figure(1);
        subplot(2,1,1);
        hold on
        plot(ExaminedRetX,Subtracted);
        if ~isempty(NewPeaks)
            scatter(NewPeaks(:,1),NewPeaks(:,2));
        end
```

```
        if ~isempty(PeakDividers)
            for j=1:PeakNum
                NumPkDiv = size(PeakDividers{j},1);
                for k=1:NumPkDiv
                    Start = PeakDividers{j}(k,1);
                    End = PeakDividers{j}(k,2);
                    scatter(ExaminedRetX(Start),Subtracted(Start),'>r','filled');
                    scatter(ExaminedRetX(End),Subtracted(End),'<r','filled');
                end
            end
        end
        if ~isempty(PeakSlopes)
            for j=1:s
                plot(PeakSlopes{j}(:,1),PeakSlopes{j}(:,2),'--r','LineWidth',1);
            end
        end
        yline(MeanSub);
        yline(Threshold);
        FixFigure(1)
        subplot(2,1,2);
        plot(ExaminedRetX(1:end-DiffWindow),dF);
        yline(MeanDer-3.5*StdDer);
        pause(1);
        close(fig1);
        hold off
    end
    end
end

w = w+1;
end

fprintf('Out of the given set of curves there are %i subpeaks\n',f);
fprintf('The interaction probability for %i curves is %d\n',w,(f/w));

if ~isempty(PeakRuptLen)
    MasterMaxForce(:,1) = PeakRuptLen';
    MasterMaxForce(:,2) = RelPeakMax';
    MasterMaxForce(:,3) = PeakArea';
    MasterMaxForce(:,4) = SlopeCoeff';
    MasterMaxForce(:,5) = PeakWidth';
    MasterMaxForce(:,6) = AbsPeakMax';
    MasterMaxForce(:,7) = CurveID';
    MasterMaxForce(:,8) = MainPeak';
end

%%% saving data
pause(1)
disp('Data analysis complete...');
pause(1)
FiletoSave = input('Please input filename (excluding dat.mat) to save the final
data: ','s');
disp('Saving...');
FiletoSave = strcat(FiletoSave,'dat','.mat');
save(FiletoSave,'MasterMaxForce','PruExt','PruRet');
pause(1)
Delte = input('Delete all raw and modified data? (1) to delete (2) to preserve: ');
if Delte==1
    if isfile('RawForceCurves.mat')
        delete('RawForceCurves.mat');
    end
    if isfile('UnprunedCurves.mat')
        delete('UnprunedCurves.mat');
    end
```

```
    if isfile('PrunedCurves.mat')
        delete('PrunedCurves.mat');
    end
    if isfile('DatabaseInfoSav.mat')
        delete('DatabaseInfoSav.mat');
    end
    if isfile('CurveData.mat')
        delete('CurveData.mat');
    end
end
disp('Exiting program...');
clear
clc
return
```

CurveQuantify – Plot extracted data into histograms and scatter plots as necessary

```
%CurveQuantify

warning('off','all');
TreeName = input('Please input the analyte name identifier: ','s');
MicaReps = input('Please input number of replicates for glass (should be equal to
mica): ');
MucinReps = input('Please input number of mucin replicates: ');
a = 1;
while a<=MicaReps
    Mica{a} = strcat(TreeName,'_Bare',num2str(a),'dat.mat');
    Mica2{a} = strcat(TreeName,'_Bare',num2str(a),'_PrePeak.mat');
    load(Mica{a});
    load(Mica2{a});
    MicaForceData{a} = [MasterMaxForce];
    MicaNumCurves{a} = [size(InflXRet,2)];
    a = a+1;
end
a = 1;
while a<=MicaReps
    Amine{a} = strcat(TreeName,'_Amine',num2str(a),'dat.mat');
    Amine2{a} = strcat(TreeName,'_Amine',num2str(a),'_PrePeak.mat');
    load(Amine{a});
    load(Amine2{a});
    AmineForceData{a} = MasterMaxForce;
    AmineNumCurves{a} = [size(InflXRet,2)];
    a = a+1;
end
a = 1;
while a<=MucinReps
    Mucin{a} = strcat(TreeName,'_Mucin',num2str(a),'dat.mat');
    Mucin2{a} = strcat(TreeName,'_Mucin',num2str(a),'_PrePeak.mat');
    load(Mucin{a});
    load(Mucin2{a});
    MucinForceData{a} = MasterMaxForce;
    MucinNumCurves{a} = [size(InflXRet,2)];
    a = a+1;
end
pause(1)
go = 1;
fign = 1;
while go==1
    proceed = input('Enter number as follows \n | (1) adhesion event probability \n
| (2) rupture length histograms \n | (3) rupture force histograms \n | (4) adhesion
energy histograms \n | (5) length-force scatter plot \n | (6) quit \n','s');
    switch proceed
        case '1'
            distancecutoff = input('Enter distance cutoff from 0-2000 nm: \n');
            distancecutoff = distancecutoff/(1000000000);
```

```
            %Adhesion event probability%
            %For Mica, Amine, and Mucin - calculate total number of curves
            %Using totally random generator, generate 20 random numbers
            %from 1 to total number of curves
            %Properly index those random numbers
            %Look up curve number in "___ForceData{i}(:,7)". Number of hits
            %is the number of subpeaks, if there are none, 0 subpeaks
            %Divide number of subpeaks by total number (20 curves)

            %Mica
            for a=1:MicaReps
                MicaTotals(a) = MicaNumCurves{a};
            end
            SumMicaTotals = sum(MicaTotals);
            for n=1:5
                rng('shuffle');
                TestRands = randi(SumMicaTotals,20,1);
                NumberSubPeaks = 0;
                for i=1:20
                    r = 0;
                    RunningTotal = 0;
                    chk = 0;
                    while (chk~=1)&&r<=MicaReps
                        r = r+1;
                        PrevTotal = RunningTotal;
                        RunningTotal = RunningTotal + MicaTotals(r);
                        if TestRands(i)<=RunningTotal
                            BatchID(i) = r;
                            Number = TestRands(i)-PrevTotal;
                            chk = 1;
                        end
                    end
                    if ~isempty(MicaForceData{BatchID(i)})
                        Tmp =
(MicaForceData{BatchID(i)}(:,7)==Number).*(MicaForceData{BatchID(i)}(:,1)>distancec
utoff);
                        NumberSubPeaks = NumberSubPeaks + sum(Tmp);
                    else
                        NumberSubPeaks = NumberSubPeaks + 0;
                    end
                end
                EventProb(n) = (NumberSubPeaks)/20;
            end
            fprintf('Mica event mean probability is %d for 5 random samples of 20
peaks\n', mean(EventProb));
            fprintf('Standard deviation of event probability is %d \n\n',
std(EventProb));

            %Amine
            for a=1:MicaReps
                AmineTotals(a) = AmineNumCurves{a};
            end
            SumAmineTotals = sum(AmineTotals);
            for n=1:5
                rng('shuffle');
                TestRands = randi(SumAmineTotals,20,1);
                NumberSubPeaks = 0;
                for i=1:20
                    r = 0;
                    RunningTotal = 0;
                    chk = 0;
                    while (chk~=1)&&r<=MicaReps
                        r = r+1;
                        PrevTotal = RunningTotal;
```

```
                        RunningTotal = RunningTotal + AmineTotals(r);
                        if TestRands(i)<=RunningTotal
                            BatchID(i) = r;
                            Number = TestRands(i)-PrevTotal;
                            chk = 1;
                        end
                    end
                    if ~isempty(AmineForceData{BatchID(i)})
                        Tmp =
(AmineForceData{BatchID(i)}(:,7)==Number).*(AmineForceData{BatchID(i)}(:,1)>distanc
ecutoff);
                        NumberSubPeaks = NumberSubPeaks + sum(Tmp);
                    else
                        NumberSubPeaks = NumberSubPeaks + 0;
                    end
                end
                EventProb(n) = (NumberSubPeaks)/20;
            end
            fprintf('Amine event mean probability is %d for 5 random samples of 20
peaks\n', mean(EventProb));
            fprintf('Standard deviation of event probability is %d \n\n',
std(EventProb));

            %Mucin
            for a=1:MucinReps
                MucinTotals(a) = MucinNumCurves{a};
            end
            SumMucinTotals = sum(MucinTotals);
            for n=1:5
                rng('shuffle');
                TestRands = randi(SumMucinTotals,20,1);
                NumberSubPeaks = 0;
                for i=1:20
                    r = 0;
                    RunningTotal = 0;
                    chk = 0;
                    while (chk~=1)&&r<=MucinReps
                        r = r+1;
                        PrevTotal = RunningTotal;
                        RunningTotal = RunningTotal + MucinTotals(r);
                        if TestRands(i)<=RunningTotal
                            BatchID(i) = r;
                            Number = TestRands(i)-PrevTotal;
                            chk = 1;
                        end
                    end
                    if ~isempty(MucinForceData{BatchID(i)})
                        Tmp =
(MucinForceData{BatchID(i)}(:,7)==Number).*(MucinForceData{BatchID(i)}(:,1)>distanc
ecutoff);
                        NumberSubPeaks = NumberSubPeaks + sum(Tmp);
                    else
                        NumberSubPeaks = NumberSubPeaks + 0;
                    end
                end
                EventProb(n) = (NumberSubPeaks)/20;
            end
            fprintf('Mucin event mean probability is %d for 5 random samples of 20
peaks\n', mean(EventProb));
            fprintf('Standard deviation of event probability is %d \n\n',
std(EventProb));


        case '2'
```

```
            %Rupture Length Histograms
            %Filter Rupture Length column to be above distance cutoff
            %Remove zero values, then add to a master 1D vector with
            %rupture lengths
            %Histograms of Rupture Lengths with Multimodal distribution
            %Plot Histogram and then Distribution on Top
            distancecutoff = input('Enter distance cutoff from 0-2000 nm: \n');
            distancecutoff = distancecutoff/(1000000000);
            LenBins = 20:20:2000;
            RupLenFig = figure(fign);
            fign = fign+1;
            %Mica
            i = 1;
            if ~isempty(MicaForceData{i})
                MicaMasterRupLen = [];
                for i=1:MicaReps

                    TmpRupLenVec = MicaForceData{i}(:,1);
                    TmpRupLenVec =
1000000000.*TmpRupLenVec.*(MicaForceData{i}(:,1)>distancecutoff);
                    TmpRupLenVec(TmpRupLenVec==0)=[];
                    MicaMasterRupLen = [MicaMasterRupLen;TmpRupLenVec];
                end
                h1 = subplot(1,3,1);
                MicaHist = histogram(MicaMasterRupLen,LenBins,'FaceColor',[0.957
0.674 0.270]);
                MicaHist.Normalization = 'count';
                ScalingFactor = sum((MicaHist.Values).*20);
                FixFigure(h1);
                xlabel('Rupture Length (nm)');
                ylabel('Counts');
                ylim(h1,[0 80]);
                if length(MicaMasterRupLen)>5
                    MicaGMModel =
fitgmdist(MicaMasterRupLen,2,'RegularizationValue',0.01);
                    MicaMeans = MicaGMModel.mu;
                    MicaComProp = MicaGMModel.ComponentProportion;
                    MicaLenPdf = pdf(MicaGMModel,LenBins');
                    plot(LenBins',MicaLenPdf.*ScalingFactor,'-k','LineWidth',1);
                    fprintf('Means of Mica Mixed Model are %d and %d
\n',MicaMeans(1),MicaMeans(2));
                    fprintf('Mixing proportions are %d and %d, respectively
\n',MicaComProp(1),MicaComProp(2));
                end
            end
            %Amine
            AmineMasterRupLen = [];
            for i=1:MicaReps
                TmpRupLenVec = AmineForceData{i}(:,1);
                TmpRupLenVec =
1000000000.*TmpRupLenVec.*(AmineForceData{i}(:,1)>distancecutoff);
                TmpRupLenVec(TmpRupLenVec==0)=[];
                AmineMasterRupLen = [AmineMasterRupLen;TmpRupLenVec];
            end
            h2 = subplot(1,3,2);
            AmineHist = histogram(AmineMasterRupLen,LenBins,'FaceColor',[0.651
0.110 0.235]);
            AmineHist.Normalization = 'count';
            ScalingFactor = sum((AmineHist.Values).*20);
            FixFigure(h2);
            xlabel('Rupture Length (nm)');
            ylabel('Counts');
            ylim(h2,[0 80]);
            if length(AmineMasterRupLen)>5
```

```
                AmineGMModel =
fitgmdist(AmineMasterRupLen,2,'RegularizationValue',0.01);
                AmineMeans = AmineGMModel.mu;
                AmineComProp = AmineGMModel.ComponentProportion;
                AmineLenPdf = pdf(AmineGMModel,LenBins');
                plot(LenBins',AmineLenPdf.*ScalingFactor,'-k','LineWidth',1);
                fprintf('Means of Amine Mixed Model are %d and %d
\n',AmineMeans(1),AmineMeans(2));
                fprintf('Mixing proportions are %d and %d, respectively
\n',AmineComProp(1),AmineComProp(2));
            end

            %Mucin
            MucinMasterRupLen = [];
            for i=1:MucinReps
                TmpRupLenVec = MucinForceData{i}(:,1);
                TmpRupLenVec =
1000000000.*TmpRupLenVec.*(MucinForceData{i}(:,1)>distancecutoff);
                TmpRupLenVec(TmpRupLenVec==0)=[];
                MucinMasterRupLen = [MucinMasterRupLen;TmpRupLenVec];
            end
            h3 = subplot(1,3,3);
            hold on
            MucinHist = histogram(MucinMasterRupLen,LenBins,'FaceColor',[0.627
0.910 0.686]);
            MucinHist.Normalization = 'count';
            ScalingFactor = sum((MucinHist.Values).*20);
            FixFigure(h3);
            xlabel('Rupture Length (nm)');
            ylabel('Counts');
            ylim(h3,[0 80]);
            xlim(h3,[0 1000]);
            if length(MucinMasterRupLen)>5
                MucinGMModel =
fitgmdist(MucinMasterRupLen,2,'RegularizationValue',0.01);
                MucinMeans = MucinGMModel.mu;
                MucinComProp = MucinGMModel.ComponentProportion;
                MucinLenPdf = pdf(MucinGMModel,LenBins');
                plot(LenBins',MucinLenPdf.*ScalingFactor,'-k','LineWidth',1);
                fprintf('Means of Mucin Mixed Model are %d and %d
\n',MucinMeans(1),MucinMeans(2));
                fprintf('Mixing proportions are %d and %d, respectively
\n',MucinComProp(1),MucinComProp(2));
                %[MucinLenLogNFit,MucinLenLogNCI] = lognfit(MucinMasterRupLen);
                %fprintf('Mean of Log Normal distribution is %d with CI of %d to %d
\n',MucinLenLogNFit(1),MucinLenLogNCI(1,1),MucinLenLogNCI(2,1));
                %fprintf('Stdev of Log Normal distribution is %d with CI of %d
to %d \n',MucinLenLogNFit(2),MucinLenLogNCI(1,2),MucinLenLogNCI(2,2));
            end
            hold off

            %Export figure as is to pdf
            %Pause to allow user to resize
            wait = input('Press any key when graphs have been appropriately
resized\n','s');
            export_fig(h3, strcat(TreeName,'_MucinRupLenDist'), '-pdf', '-
nofontswap');

        case '3'
            %Rupture force Histograms
            %Filter Rupture force column to be above distance cutoff
            %Remove zero values, then add to a master 1D vector with
            %rupture forces
            %Histograms of Rupture forces with Multimodal distribution
```

```
            %Plot Histogram and then Distribution on Top
            distancecutoff = input('Enter distance cutoff from 0-2000 nm: \n');
            distancecutoff = distancecutoff/(1000000000);
            ForBins = 10:10:600;
            RupForFig = figure(fign);
            fign = fign + 1;
            %Mica
            i = 1;
            if ~isempty(MicaForceData{i})
                MicaMasterRupFor = [];
                for i=1:MicaReps

                    TmpRupForVec = MicaForceData{i}(:,2);
                    TmpRupForVec =
1000000000.*TmpRupForVec.*(MicaForceData{i}(:,1)>distancecutoff);
                    TmpRupForVec(TmpRupForVec==0)=[];
                    MicaMasterRupFor = [MicaMasterRupFor;TmpRupForVec];
                end
                h1 = subplot(1,3,1);
                MicaHist = histogram(MicaMasterRupFor,ForBins,'FaceColor',[0.957
0.674 0.270]);
                MicaHist.Normalization = 'count';
                ScalingFactor = sum((MicaHist.Values).*10);
                FixFigure(h1);
                xlabel('Rupture Length (nm)');
                ylabel('Counts');
                ylim(h1,[0 80]);
                if length(MicaMasterRupFor)>5
                    MicaGMModel =
fitgmdist(MicaMasterRupFor,2,'RegularizationValue',0.01);
                    MicaMeans = MicaGMModel.mu;
                    MicaComProp = MicaGMModel.ComponentProportion;
                    MicaForPdf = pdf(MicaGMModel,ForBins');
                    plot(ForBins',MicaForPdf.*ScalingFactor,'-k','LineWidth',1);
                    fprintf('Means of Mica Mixed Model are %d and %d
\n',MicaMeans(1),MicaMeans(2));
                    fprintf('Mixing proportions are %d and %d, respectively
\n',MicaComProp(1),MicaComProp(2));

                end
            end
            %Amine
            AmineMasterRupFor = [];
            for i=1:MicaReps
                TmpRupForVec = AmineForceData{i}(:,2);
                TmpRupForVec =
1000000000000.*TmpRupForVec.*(AmineForceData{i}(:,1)>distancecutoff);
                TmpRupForVec(TmpRupForVec==0)=[];
                AmineMasterRupFor = [AmineMasterRupFor;TmpRupForVec];
            end
            h2 = subplot(1,3,2);
            AmineHist = histogram(AmineMasterRupFor,ForBins,'FaceColor',[0.651
0.110 0.235]);
            AmineHist.Normalization = 'count';
            ScalingFactor = sum((AmineHist.Values).*10);
            FixFigure(h2);
            xlabel('Rupture Length (nm)');
            ylabel('Counts');
            ylim(h2,[0 80]);
            if length(AmineMasterRupFor)>5
                AmineGMModel =
fitgmdist(AmineMasterRupFor,2,'RegularizationValue',0.01);
                AmineMeans = AmineGMModel.mu;
                AmineComProp = AmineGMModel.ComponentProportion;
```

```
                AmineForPdf = pdf(AmineGMModel,ForBins');
                plot(ForBins',AmineForPdf.*ScalingFactor,'-k','LineWidth',1);
                fprintf('Means of Amine Mixed Model are %d and %d
\n',AmineMeans(1),AmineMeans(2));
                fprintf('Mixing proportions are %d and %d, respectively
\n',AmineComProp(1),AmineComProp(2));
            end

            %Mucin
            MucinMasterForLen = [];
            for i=1:MucinReps
                TmpRupForVec = MucinForceData{i}(:,2);
                TmpRupForVec =
1000000000000.*TmpRupForVec.*(MucinForceData{i}(:,1)>distancecutoff);
                TmpRupForVec(TmpRupForVec==0)=[];
                MucinMasterForLen = [MucinMasterForLen;TmpRupForVec];
            end
            h3 = subplot(1,3,3);
            hold on
            MucinHist = histogram(MucinMasterForLen,ForBins,'FaceColor',[0.765
0.490 0.572]);
            MucinHist.Normalization = 'count';
            ScalingFactor = sum((MucinHist.Values).*10);
            FixFigure(h3);
            xlabel('Rupture force (pN)');
            ylabel('Counts');
            ylim(h3,[0 80]);
            if length(MucinMasterForLen)>5
                MucinGMModel =
fitgmdist(MucinMasterForLen,2,'RegularizationValue',0.01);
                MucinMeans = MucinGMModel.mu;
                MucinComProp = MucinGMModel.ComponentProportion;
                MucinForPdf = pdf(MucinGMModel,ForBins');
                plot(ForBins',MucinForPdf.*ScalingFactor,'-k','LineWidth',1);
                fprintf('Means of Mucin Mixed Model are %d and %d
\n',MucinMeans(1),MucinMeans(2));
                fprintf('Mixing proportions are %d and %d, respectively
\n',MucinComProp(1),MucinComProp(2));
                [MucinForLogNFit,MucinForLogNCI] = lognfit(MucinMasterForLen);
                fprintf('Mean of Log Normal distribution is %d with CI of %d to %d
\n',MucinForLogNFit(1),MucinForLogNCI(1,1),MucinForLogNCI(2,1));
                fprintf('Stdev of Log Normal distribution is %d with CI of %d to %d
\n',MucinForLogNFit(2),MucinForLogNCI(1,2),MucinForLogNCI(2,2));
            end
            hold off

            %Export figure as is to pdf
            %Pause to allow user to resize
            wait = input('Press any key when graphs have been appropriately
resized\n','s');
            export_fig(h3, strcat(TreeName,'_MucinRupForDist'), '-pdf', '-
nofontswap');

        case '4'
            %Adhesion Energy Histograms
            %Filter Adhesion Energy column to be above distance cutoff
            %Remove zero values, then add to a master 1D vector with
            %rupture lengths
            %Histograms of Adhesion Energy with Multimodal distribution
            %Plot Histogram and then Distribution on Top
            distancecutoff = input('Enter distance cutoff from 0-2000 nm: \n');
            distancecutoff = distancecutoff/(1000000000);
            EnBins = 0.5:3:150;
            RupEnFig = figure(fign);
```

```
            fign = fign+1;
            %Mica
            i = 1;
            if ~isempty(MicaForceData{i})
                MicaMasterRupEn = [];
                for i=1:MicaReps
                    TmpRupEnVec = MicaForceData{i}(:,3);
                    TmpRupEnVec =
10000000000000000.*TmpRupEnVec.*(MicaForceData{i}(:,1)>distancecutoff);
                    TmpRupEnVec(TmpRupEnVec<=0)=[];
                    MicaMasterRupEn = [MicaMasterRupEn;TmpRupEnVec];
                end
                h1 = subplot(1,3,1);
                MicaHist = histogram(MicaMasterRupEn,EnBins,'FaceColor',[0.957
0.674 0.270]);
                MicaHist.Normalization = 'count';
                ScalingFactor = sum((MicaHist.Values).*3);
                FixFigure(h1);
                xlabel('Adhesion energy (aJ)');
                ylabel('Counts');
                ylim(h1,[0 150]);
                if length(MicaMasterRupEn)>5
                    MicaGMModel =
fitgmdist(MicaMasterRupEn,2,'RegularizationValue',0.01);
                    MicaMeans = MicaGMModel.mu;
                    MicaComProp = MicaGMModel.ComponentProportion;
                    MicaEnPdf = pdf(MicaGMModel,EnBins');
                    plot(EnBins',MicaEnPdf.*ScalingFactor,'-k','LineWidth',1);
                    fprintf('Means of Mica Mixed Model are %d and %d
\n',MicaMeans(1),MicaMeans(2));
                    fprintf('Mixing proportions are %d and %d, respectively
\n',MicaComProp(1),MicaComProp(2));
                end
            end
            %Amine
            AmineMasterRupEn = [];
            for i=1:MicaReps
                TmpRupEnVec = AmineForceData{i}(:,3);
                TmpRupEnVec =
10000000000000000.*TmpRupEnVec.*(AmineForceData{i}(:,1)>distancecutoff);
                TmpRupEnVec(TmpRupEnVec<=0)=[];
                AmineMasterRupEn = [AmineMasterRupEn;TmpRupEnVec];
            end
            h2 = subplot(1,3,2);
            AmineHist = histogram(AmineMasterRupEn,EnBins,'FaceColor',[0.651 0.110
0.235]);
            AmineHist.Normalization = 'count';
            ScalingFactor = sum((AmineHist.Values).*3);
            FixFigure(h2);
            xlabel('Adhesion energy (aJ)');
            ylabel('Counts');
            ylim(h2,[0 150]);
            if length(AmineMasterRupEn)>5
                AmineGMModel =
fitgmdist(AmineMasterRupEn,2,'RegularizationValue',0.01);
                AmineMeans = AmineGMModel.mu;
                AmineComProp = AmineGMModel.ComponentProportion;
                AmineEnPdf = pdf(AmineGMModel,EnBins');
                plot(EnBins',AmineEnPdf.*ScalingFactor,'-k','LineWidth',1);
                fprintf('Means of Amine Mixed Model are %d and %d
\n',AmineMeans(1),AmineMeans(2));
                fprintf('Mixing proportions are %d and %d, respectively
\n',AmineComProp(1),AmineComProp(2));
            end
```

```
            %Mucin
            MucinMasterEnLen = [];
            for i=1:MucinReps
                TmpRupEnVec = MucinForceData{i}(:,3);
                TmpRupEnVec =
10000000000000000.*TmpRupEnVec.*(MucinForceData{i}(:,1)>distancecutoff);
                TmpRupEnVec(TmpRupEnVec<=0)=[];
                MucinMasterEnLen = [MucinMasterEnLen;TmpRupEnVec];
            end
            h3 = subplot(1,3,3);
            hold on
            MucinHist = histogram(MucinMasterEnLen,EnBins,'FaceColor',[0.474 0.745
0.933]);
            MucinHist.Normalization = 'count';
            ScalingFactor = sum((MucinHist.Values).*3);
            FixFigure(h3);
            xlabel('Adhesion energy (aJ)');
            ylabel('Counts');
            ylim(h3,[0 150]);
            if length(MucinMasterEnLen)>5
                MucinGMModel =
fitgmdist(MucinMasterEnLen,2,'RegularizationValue',0.01);
                MucinMeans = MucinGMModel.mu;
                MucinComProp = MucinGMModel.ComponentProportion;
                MucinEnPdf = pdf(MucinGMModel,EnBins');
                plot(EnBins',MucinEnPdf.*ScalingFactor,'-k','LineWidth',1);
                fprintf('Means of Mucin Mixed Model are %d and %d
\n',MucinMeans(1),MucinMeans(2));
                fprintf('Mixing proportions are %d and %d, respectively
\n',MucinComProp(1),MucinComProp(2));
                [MucinEnLogNFit,MucinEnLogNCI] = lognfit(MucinMasterEnLen);
                fprintf('Mean of Log Normal distribution is %d with CI of %d to %d
\n',MucinEnLogNFit(1),MucinEnLogNCI(1,1),MucinEnLogNCI(2,1));
                fprintf('Stdev of Log Normal distribution is %d with CI of %d to %d
\n',MucinEnLogNFit(2),MucinEnLogNCI(1,2),MucinEnLogNCI(2,2));
            end
            hold off

            %Export figure as is to pdf
            %Pause to allow user to resize
            wait = input('Press any key when graphs have been appropriately
resized\n','s');
            export_fig(h3, strcat(TreeName,'_MucinRupEnDist'), '-pdf', '-
nofontswap');

        case '5'
            %Force v Length v Energy Scatter plot
            distancecutoff = input('Enter distance cutoff from 0-2000 nm: \n');
            distancecutoff = distancecutoff/(1000000000);

            %Load all rupture lengths for Mucin only into Vector
            MucinMasterRupLen = [];
            for i=1:MucinReps
                TmpRupLenVec = MucinForceData{i}(:,1);
                TmpRupLenVec =
1000000000.*TmpRupLenVec.*(MucinForceData{i}(:,1)>distancecutoff);
                MucinMasterRupLen = [MucinMasterRupLen;TmpRupLenVec];
            end
            %Load all rupture forces for Mucin only into Vector
            MucinMasterForLen = [];
            for i=1:MucinReps
                TmpRupForVec = MucinForceData{i}(:,2);
```

```
                TmpRupForVec =
1000000000000.*TmpRupForVec.*(MucinForceData{i}(:,1)>distancecutoff);
                MucinMasterForLen = [MucinMasterForLen;TmpRupForVec];
            end
            %Load all adhesion energies for Mucin only into vector
            %Want size vector such that mean adhesion energy for AmPAS50
            %yields a size of 50
            MucinMasterEnLen = [];
            for i=1:MucinReps
                TmpRupEnVec = MucinForceData{i}(:,3);
                TmpRupEnVec =
10000000000000000.*TmpRupEnVec.*(MucinForceData{i}(:,1)>distancecutoff);
                MucinMasterEnLen = [MucinMasterEnLen;TmpRupEnVec];
            end
            DimCheck =
(MucinMasterRupLen>0).*(MucinMasterForLen>0).*(MucinMasterEnLen>0);
            MucinMasterRupLen = MucinMasterRupLen.*DimCheck;
            MucinMasterForLen = MucinMasterForLen.*DimCheck;
            MucinMasterEnLen = MucinMasterEnLen.*DimCheck;
            MucinMasterRupLen(MucinMasterRupLen<=0) = [];
            MucinMasterForLen(MucinMasterForLen<=0) = [];
            MucinMasterEnLen(MucinMasterEnLen<=0) = [];
            Avg = 12.1594; %Variable depending on what the highest interaction
partner's mean is
            SizeVec = 15.*(MucinMasterEnLen./Avg);
            f2 = figure(fign);
            fign = fign + 1;
            hold on
            h5 = subplot(1,1,1);
            scatter(MucinMasterRupLen,MucinMasterForLen,SizeVec,[1 0.698
0.220],'filled');
            a2 = gca;
            FixFigure(a2);
            xlim([0 1500]);
            ylim([0 600]);
            xlabel('Rupture length (nm)');
            ylabel('Rupture force (pN)');
%             h6 = subplot(1,2,1);
%             scatter(MucinMasterRupLen,MucinMasterEnLen,'k','filled');
%             a2 = gca;
%             FixFigure(a2);
%             xlim([0 1500]);
%             ylim([0 100]);
%             xlabel('Rupture length (nm)');
%             ylabel('Adhesion energy (aJ)');
%             hold off

            %Export figure as is to pdf
            %Pause to allow user to resize
            wait = input('Press any key when graphs have been appropriately
resized\n','s');
            export_fig(h5,strcat(TreeName,'_MucinScatter'), '-pdf', '-nofontswap');
            export_fig(h6,strcat(TreeName,'_MucinEnScatter'), '-pdf', '-
nofontswap');

        case '6'
            go = 0;

        otherwise
            disp('Failed to recognize - enter only one number or (6) to quit');
            go = 1;
    end
end
clear
```

References

[1] J. P. Lellouche, S. Koeller, *J. Org. Chem.* **2001**, *66*, 693–696.

[2] H. Mikula, D. Matscheko, M. Schwarz, C. Hametner, J. Fröhlich, *Carbohydr. Res.* **2013**, *370*, 19–23.

[3] S. Bijland, G. Thomson, M. Euston, K. Michail, K. Thümmler, S. Mücklisch, C. L. Crawford, S. C. Barnett, M. McLaughlin, T. J. Anderson, et al., *F1000Research* **2019**, *8*, 117.

[4] R. K. Jain, R. J. Stock, S. R. Chary, M. Rueter, *Microvasc. Res.* **1990**, *39*, 77–93.

Author Contributions

A.S.B. wrote the manuscript, performed all synthetic procedures, performed all molecular characterization, was involved in data collection for Figures 1-4 and Figures S1-20, and prepared all figures. R.C.S. performed data collection for Figures 2-4 and S12-18 and helped edit the manuscript. M.C. performed data collection for Figure S19 and helped edit the manuscript. B.S. and M.W.G. provided research direction, mentorship, and helped edit the manuscript.