Electronic Supplementary Material (ESI) for Green Chemistry.

Sustainability by Design: Automated Nanoscale 2,3,4-Trisubstituted Quinazoline Diversity

Mojgan Hadian,[1‡] Shabnam Shaabani,[1‡] Pravin Patil,[1‡] Svitlana V. Shishkina,[2] Harry Boeltz,[3] Alexander Dömling[1]*

[1] Pharmacy Department, Drug Design group, University of Groningen, Netherlands.

[2] SSI "Institute for Single Crystals," National Academy of Science of Ukraine, 60 Lenina Ave., Kharkiv 61001, Ukraine.

[3] Dispendix GmbH, Heßbrühlstraße 7, 70565 Stuttgart, Germany.

‡ These authors contributed equally to this work.

*Correspondence: a.s.s.domling@rug.nl

Table of contents

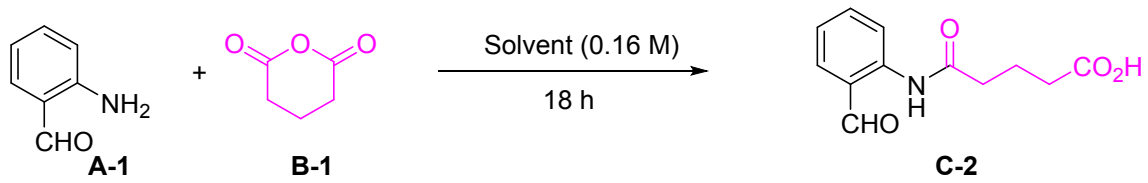
1. General information

1.1. Materials and methods

Nuclear magnetic resonance spectra (NMR) were recorded on a Bruker Avance 500 spectrometer (^{1}H NMR (500 MHz), ^{13}C NMR (126 MHz)). Chemical shifts for ^{1}H NMR were reported relative to TMS (δ 0 ppm) or internal solvent peak ($CDCl_3$ δ 7.26 ppm, DMSO-d_6 δ 2.50 ppm or CD_3OD δ 3.31 ppm) and coupling constants were in hertz (Hz). The following abbreviations were used for spin multiplicity: s = singlet, d = doublet, t = triplet, dd = double doublet, m = multiplet, brs = broad singlet. Chemical shifts for ^{13}C NMR reported in ppm relative to the solvent peak ($CDCl_3$ δ 77.00 ppm, DMSO-d_6 δ 39.50 ppm, CD_3OD δ 49.00 ppm). Thin layer chromatography was performed on precoated silica gel 60 F_{254} plates (Merck, Darmstadt). Reagents were available from commercial suppliers and used without any purification unless otherwise noted. All isocyanides were made in house by either performing the Hoffman or Ugi procedure. Other reagents were purchased from Sigma Aldrich, ABCR, Acros and AK Scientific and were used without further purification. Yields given refer to chromatographically purified and spectroscopically pure compounds unless otherwise stated. Melting points were determined using a OEM Electrothermal melting point apparatus 1A 8103. Electrospray ionization mass spectra (ESI-MS) were recorded on a Waters Investigator Semi-prep 15 SFC-MS instrument. High-resolution mass spectra were recorded using a QTOF Bruker Maxis Plus, mass range 100-1500 m/z, spectra rate 2.00 Hz. IR spectra were recorded on a Thermo Nicolet Avatar 380 FTIR with Diamond ATR.

2. Optimization of the reaction conditions

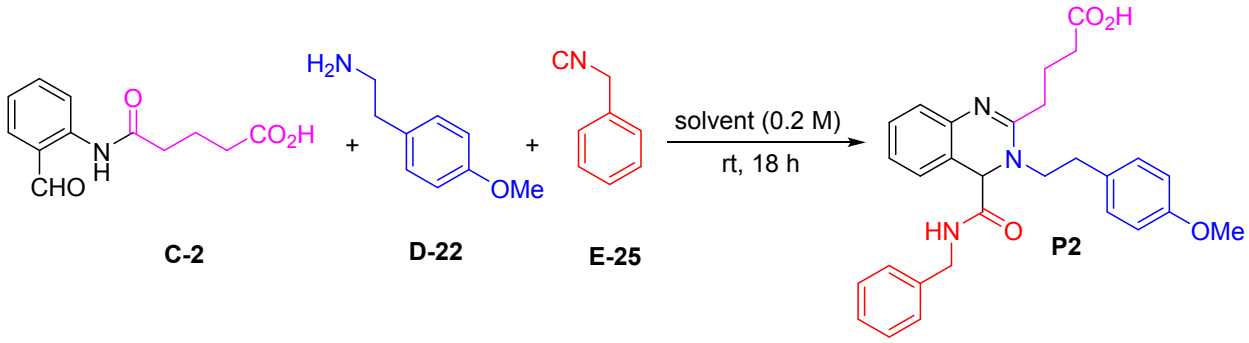
Table S1. Optimization of the α-formyl-ω-carboxylic acid synthesis[a]



Entry	A-1/B-1	Solvent	Temperature [°C]	Yield [%][b]
1	1:2	Benzene	75	30 [c]
2	1:2	Toluene	75	30 [c]
3	1:2	*n*-Heptane	75	35
4	1:2	Dioxane	75	55
5	1:2	Dioxane:*n*-Heptane (1:1)	75	45
6	1:2	Dioxane:Benzene (1:1)	75	45
7	1:1.8	Dioxane	75	55
8	1:1.5	Dioxane	75	60
9	1:1.1	Dioxane	75	70
10	1:1	Dioxane	75	60
11	1:2	Dioxane	Reflux	60
12	1:1.8	Dioxane	Reflux	60
13	1:1.5	Dioxane	Reflux	80
14	**1:1.1**	**Dioxane**	**Reflux**	**95**
15	1:1	Dioxane	Reflux	75

[a] The reactions were run on 1 mmol scale; [b] Isolated yield; [c] Yields based on SFC-MS analysis.

Table S2. Optimization of the 2,3,4-trisubstituted quinazoline synthesis[a]



Entry	C-2/D-22/E-25	Solvent	Catalyst (equiv)	Yield [%][b]
1	1:2:2	THF	-	30 [c]
2	1:2:2	CH_3CN	-	22 [c]
3	1:2:2	MeOH	-	71
4	1:2:2	TFE	-	76
5	1:2:2	TFE	$ZnCl_2$ (0.1)	75
6	1:2:2	TFE	$Sc(OTf)_3$ (0.1)	76
7	1:2:2	TFE	$Al(OTf)_3$ (0.1)	75
8	1:2:2	TFE	*p*-TSA (0.1)	72
9	1:1.5:1.5	TFE	-	79
10	**1:1.1:1.1**	**TFE**	**-**	**84**
11	1:1:1	TFE	-	65
12 [d]	1:1.1:1.1	TFE	-	55
13 [e]	1:1.1:1.1	TFE	-	74

[a] The reactions were run on 1 mmol scale; [b] Isolated yield; [c] Yields based on SFC-MS analysis; [d] Sonication (8 h); [e] MW (80 ℃, 1 h).

3. Nano-scale automated chemistry

3.1. General materials

Stock solutions were prepared in glass flat bottom vials (Screening devices, Catalog#: 9920-812FBT, 2.0 mL (Topas) Plate), and they were stored at -20 °C.

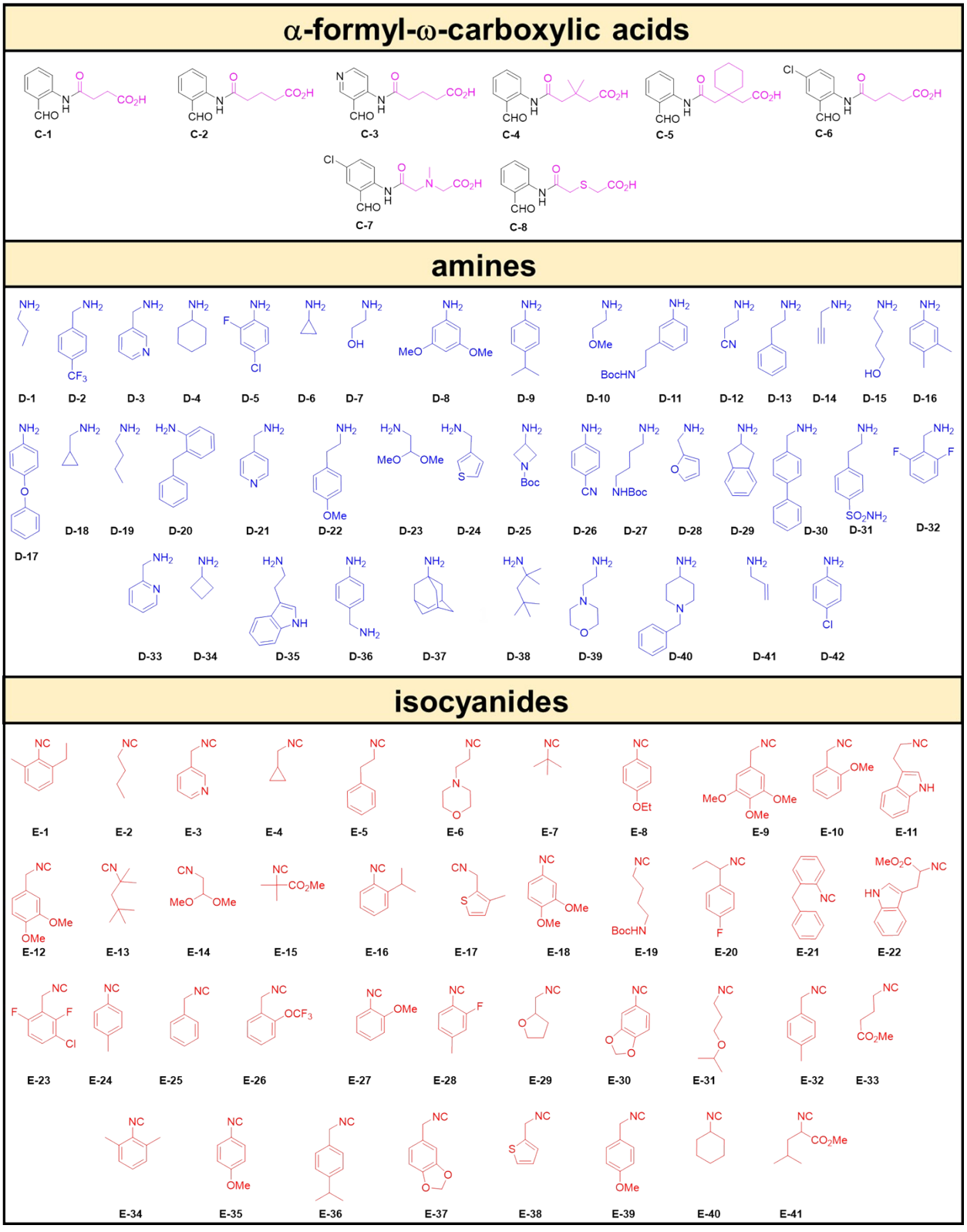
Nanomole-scale chemistry was performed using Dispendix I-DOT PURE plate, 96-well polypropylene plate, as source plate. 60 µm orifice wells (commercially not available) were used for the transfer of reagents. Greiner bio-one, 384-well PCR polypropylene plate (LOT#: E19043P9, REF#: 785290) were used as destination plate.

384-well destination plates were sealed by a sealing tape (Thermo Scientific, Catalog#: 232701, polyolefin acrylate) and were stored at -20 °C.

3.2. Instrumentation

The I-DOT One liquid handler (Dispendix) was used in order to transfer nL droplets of starting materials from the 96-well source plate to the 384-well destination plate.

3.3. Structures of the building blocks



3.4. Stock solution preparation

The stock solution of α-formyl-ω-carboxylic acids (**C-4**, **C-7**) were prepared as 0.5 M TFE. Due to the insolubility, (**C1-C3**, **C-5**) were prepared as 0.25 M TFE and (**C-6**, **C-8)** were prepared as 0.16 M TFE/water (10:1).

The stock solutions of amine (**D1-D30**, **D32-D39**, **D41-42**) were prepared as 0.5 M TFE. Due to the insolubility **D-31**, **D-40** were prepared as 0.25 M TFE.

The stock solutions of isocyanides (**E1-E41**) were prepared as 0.5 M TFE.

3.5. Nano-scale synthesis

The stock solutions were dispensed to a 96-well source plate using Eppendorf multi-channel pipettes.

The I-DOT was used to transfer the α-formyl-ω-carboxylic acids (1 eq, 163 nL), the amines (1 eq, 163 nL) and the isocyanides (1 eq, 163 nL) into the corresponding well in the destination plate. In case of diluted starting materials with 0.25 M and 0.16 M concentrations, 325 nL and 488 nL were transferred, respectively. I-DOT Assay Studio software with the pick list as a csv file was used (Fig. S1).

In order to generate a random library of products (N=384), a modified version of our previously reported program RandReactor was used.[1] The smiles files of the starting materials with the corresponding location in the source plate and mrv file of reaction were the input of the RandReactor program. The smiles file of the randomly generated products with their corresponding locations in the source and destination plate were the output of the RandReactor program. The smiles file was converted to a csv file which was the required format for I-DOT Assay Studio software (Fig. S1A).

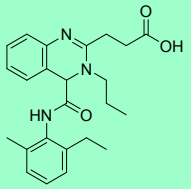
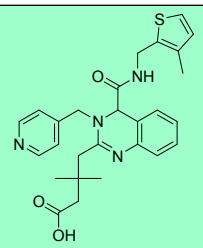
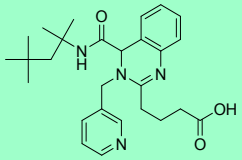
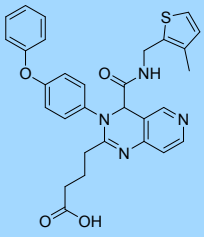
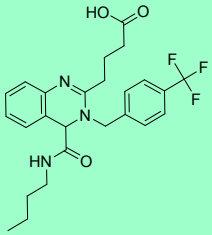
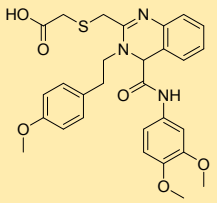
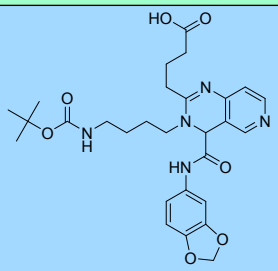
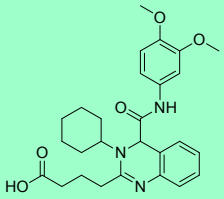
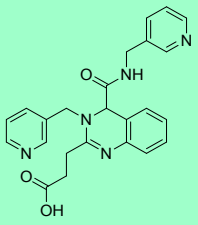
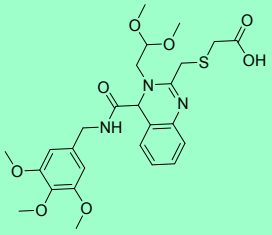
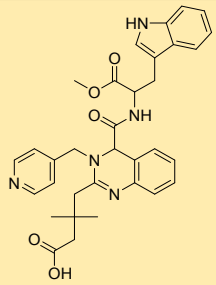
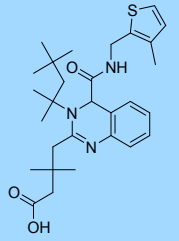
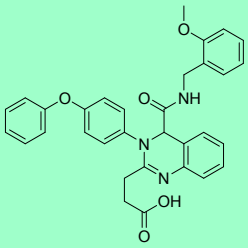
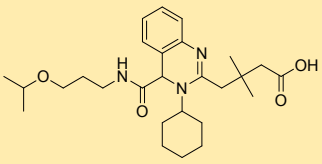
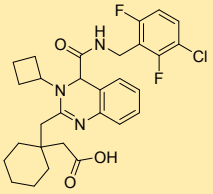
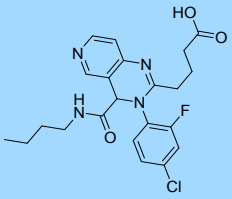
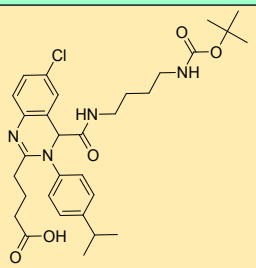
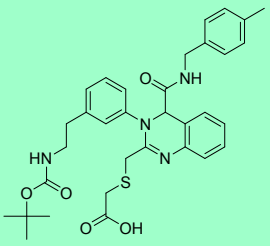
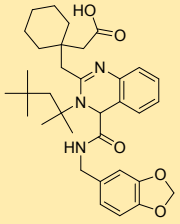
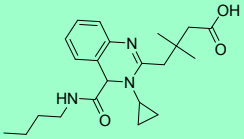
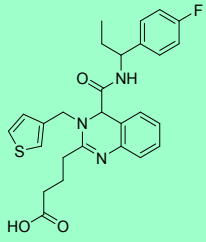
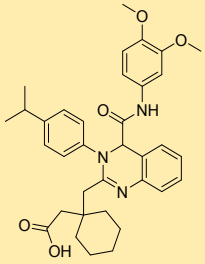
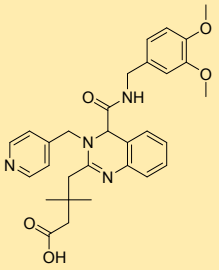
Once the starting materials transfer was completed (~15 min), the destination plate was covered with the sealing film and was then placed for 24 h at 23 °C on an orbital shaker. The sealed plates were stored at -20 °C for further processing. The structures of the products are shown in Fig. S2.

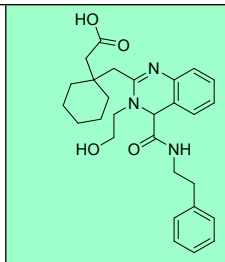
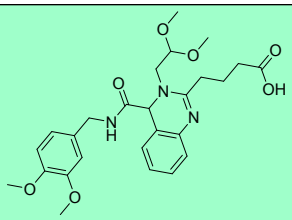
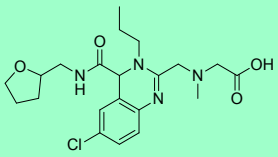
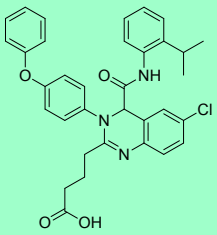
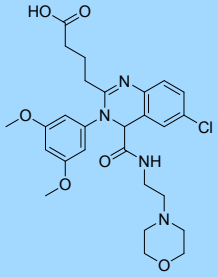
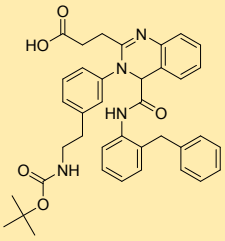
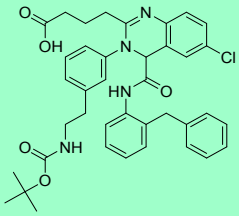
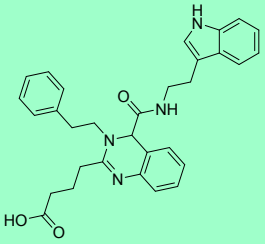
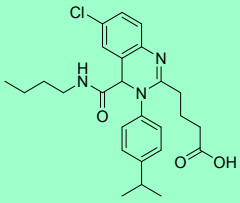
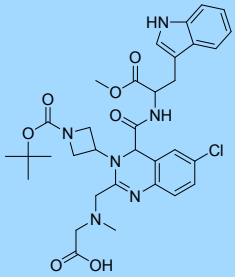
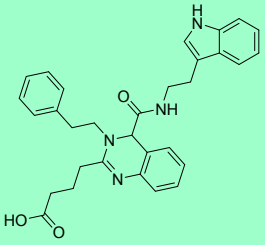
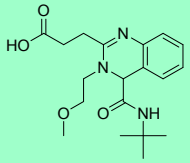
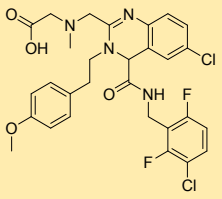
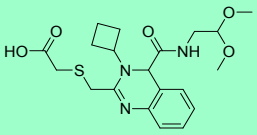
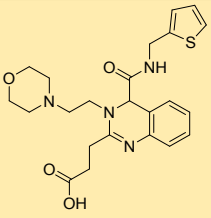
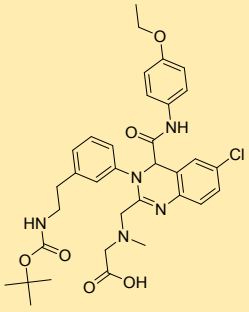
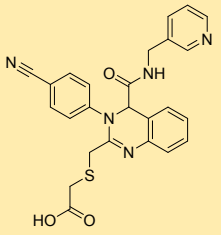
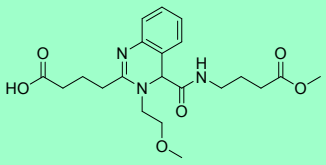
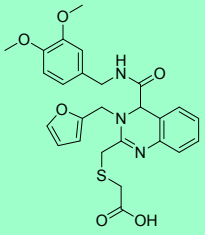
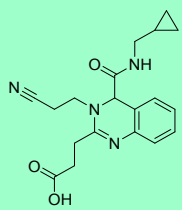
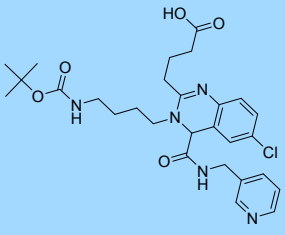
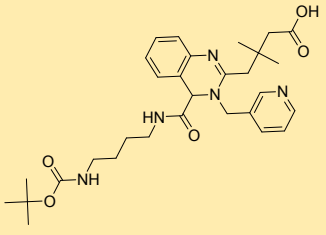
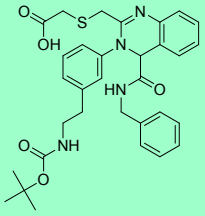
A

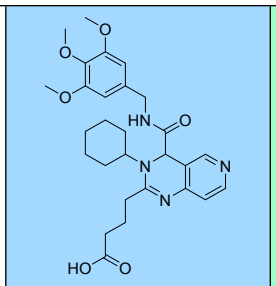
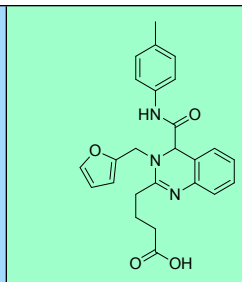
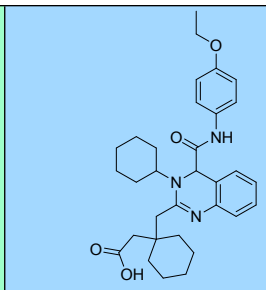
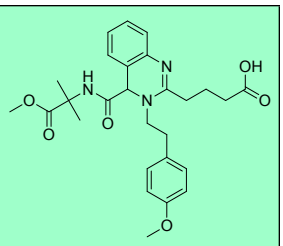
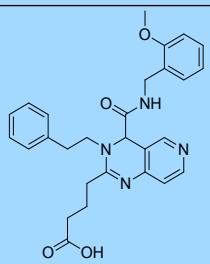
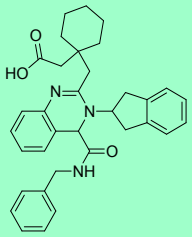
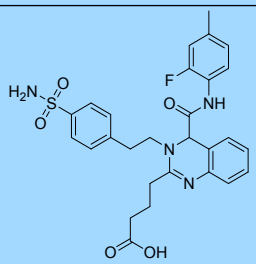
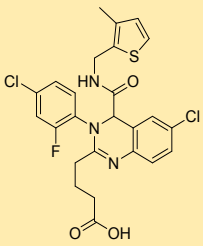
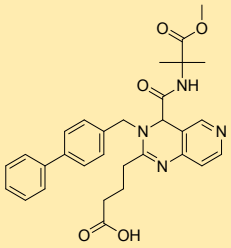
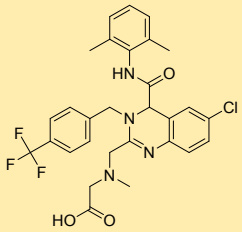
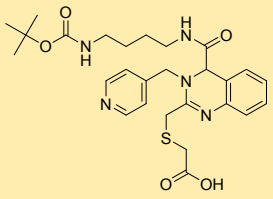
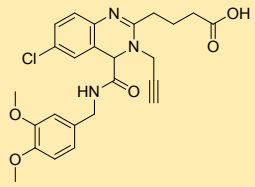
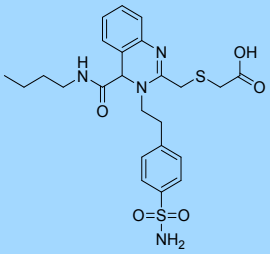
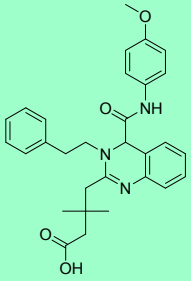
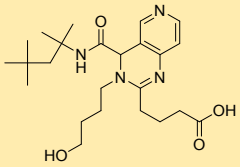
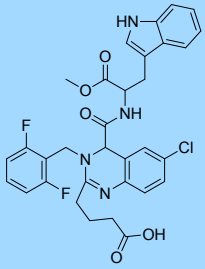
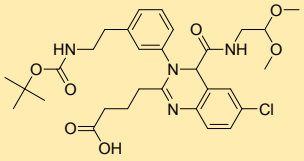
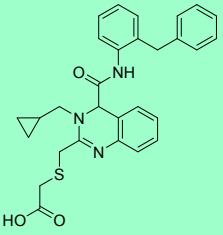
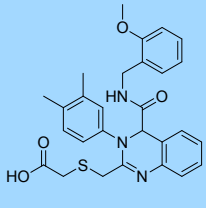
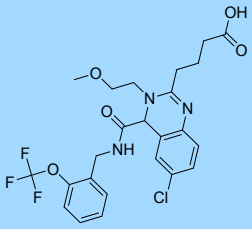
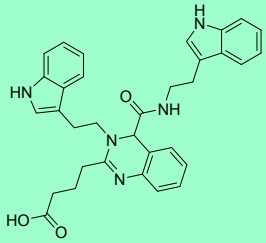
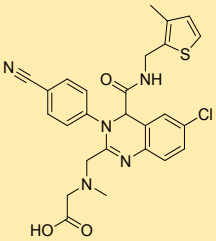
	A	B	C	D	E	F	G	H
1	User	1.2.	<User Name>	22-1-2020	12:10			
2	PurePlate90	Source Plate 1		MWP 384	Target Plate 1			
3	DispenseToWaste=False	DispenseToWasteCycles=3	DispenseToWasteVolume=1e-7	UseDeionisation=True	OptimizationLevel=ReorderAndParallel	WasteErrorHandlingLevel=Ask		
4	Source Well	Target Well	Volume [L]	Liquid Name				
5	F11	A1	1.63E-07	F11				
6	A10	A2	1.63E-07	A10				
7	H12	A3	1.63E-07	H12				
8	B8	A4	1.63E-07	B8				
9	A10	A5	1.63E-07	A10				
10	A10	A6	1.63E-07	A10				
11	D12	A7	1.63E-07	D12				
12	E9	A8	1.63E-07	E9				
13	A10	A9	1.63E-07	A10				
14	E8	A10	1.63E-07	E8				
15	B10	A11	1.63E-07	B10				
16	B8	A12	1.63E-07	B8				
17	B9	A13	1.63E-07	B9				
18	G10	A14	1.63E-07	G10				
19	H9	A15	1.63E-07	H9				
20	G9	A16	1.63E-07	G9				
21	A9	A17	1.63E-07	A9				
22	G10	A18	1.63E-07	G10				
23	H7	A19	1.63E-07	H7				

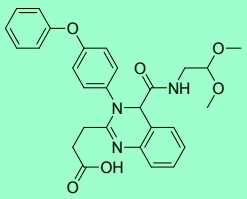
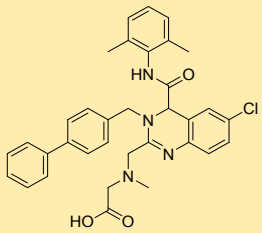
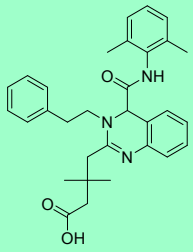
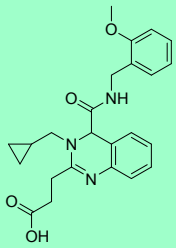
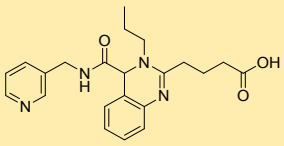
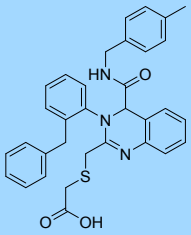
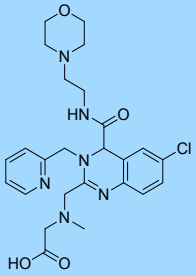
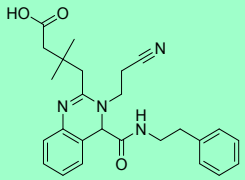
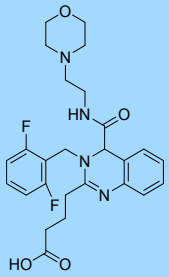
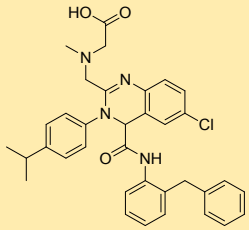
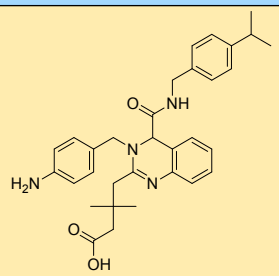
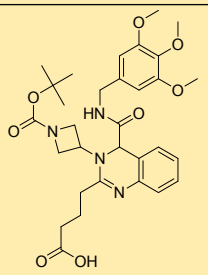
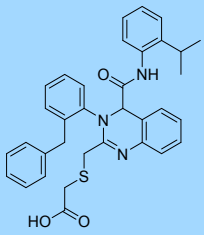
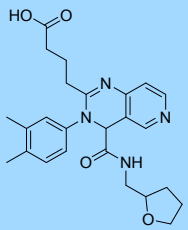
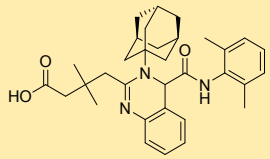
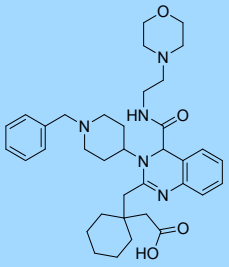
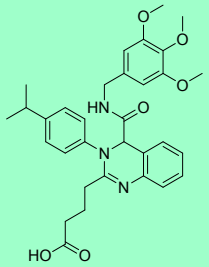
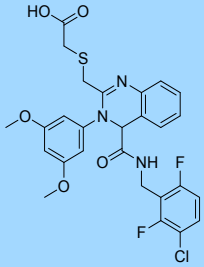
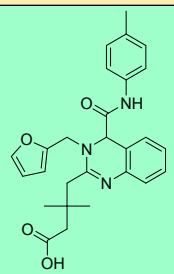
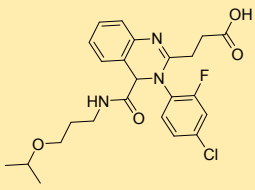
B

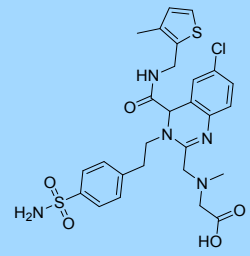
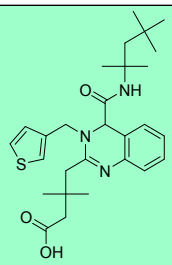
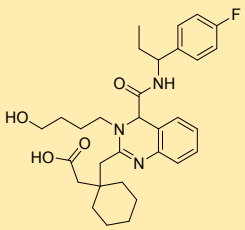
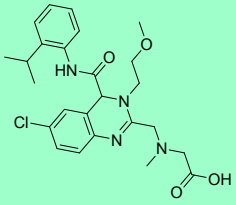
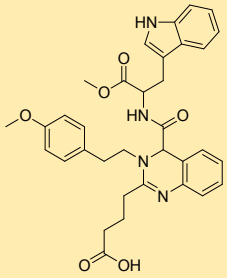
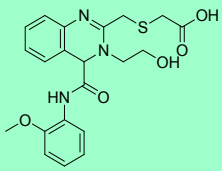
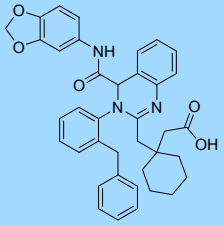
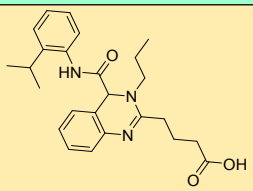
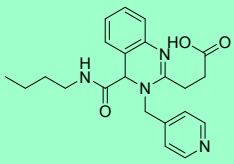
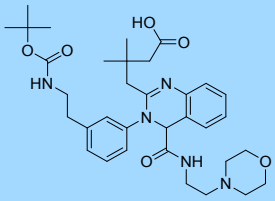
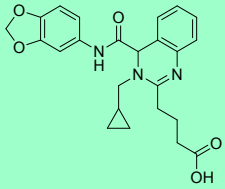
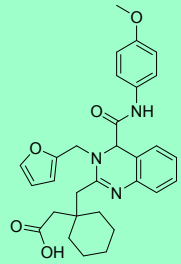
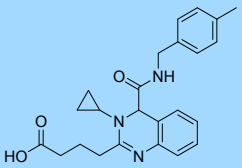
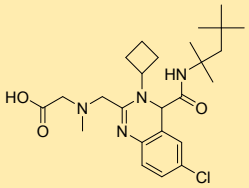
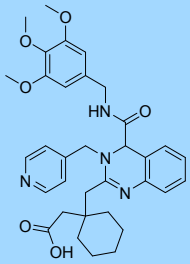
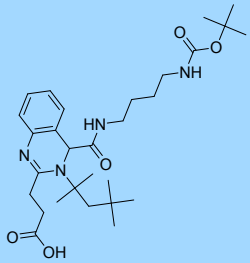
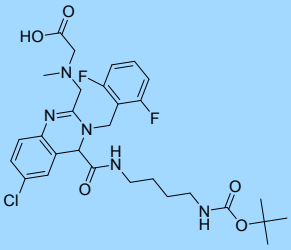
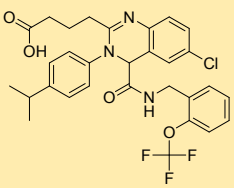
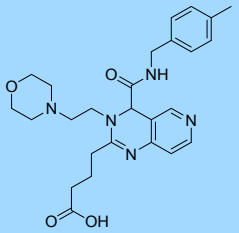
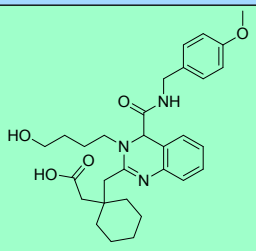
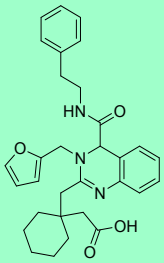
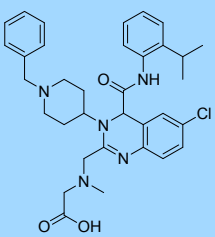
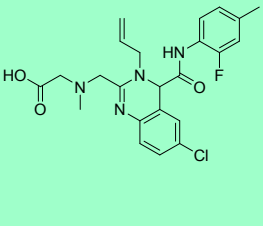
Fig. S1. A: Pick list in csv format; **B:** I-DOT Assay Studio software, showing the destination plate layout.

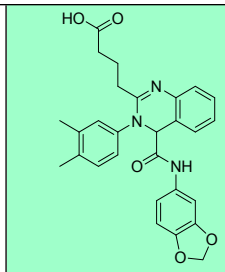
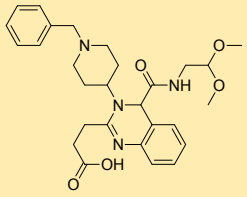
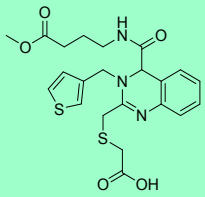
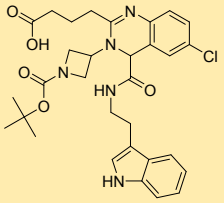
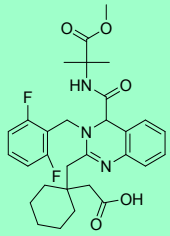
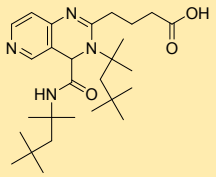
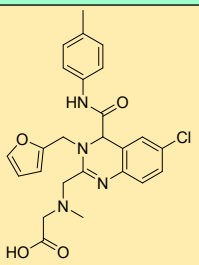
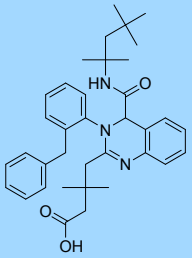
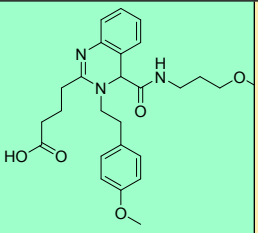
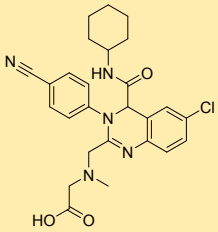
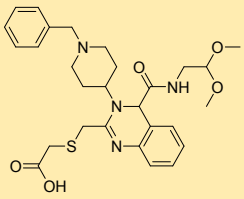
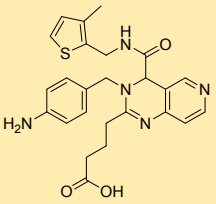
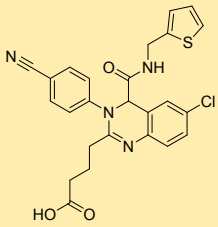
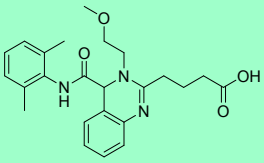
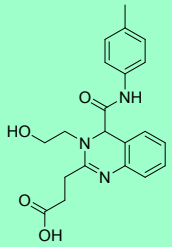
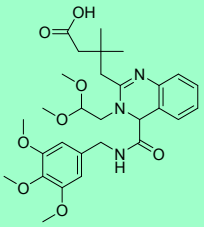
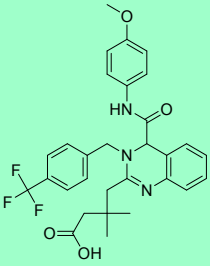
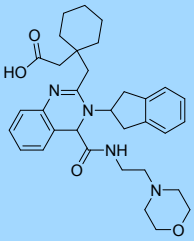
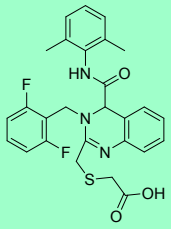
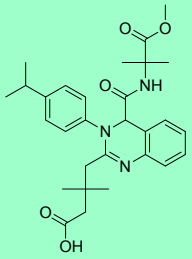
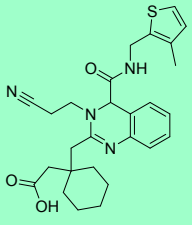
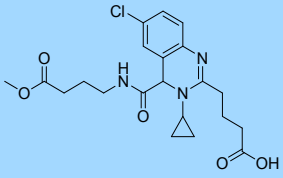
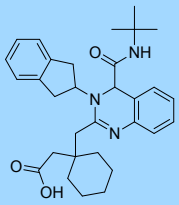
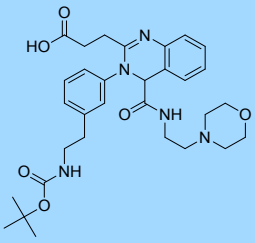
	A	B	C	D
1				
2				
3				
4				
5				
6				

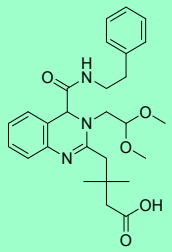
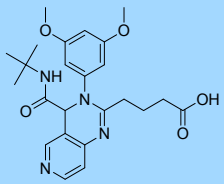
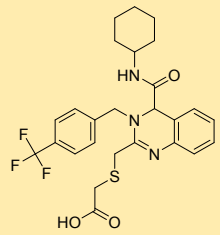
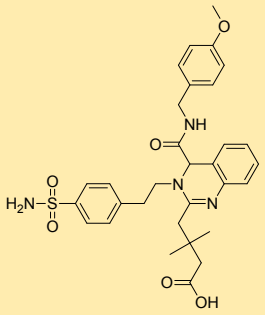
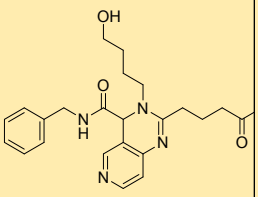
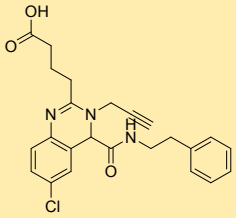
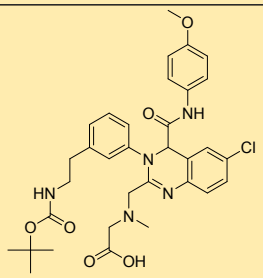
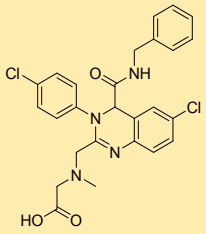
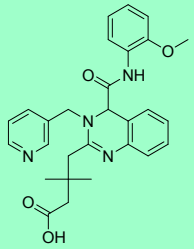
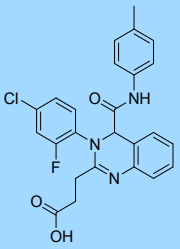
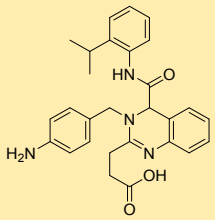
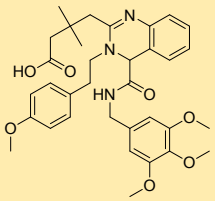
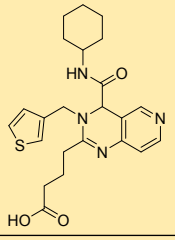
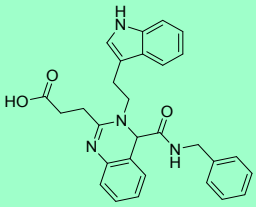
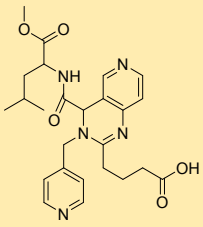
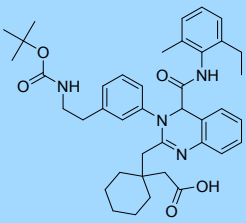
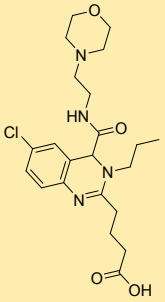
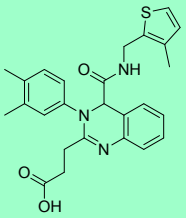
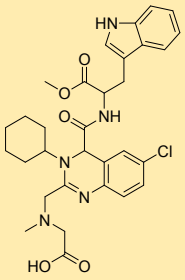
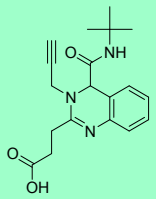
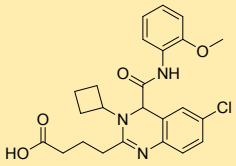
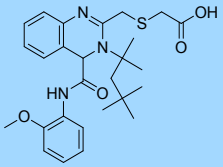
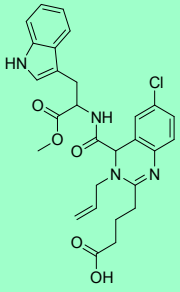
7				
8				
9				
10				
11				
12				

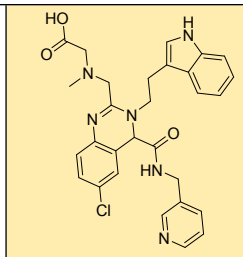
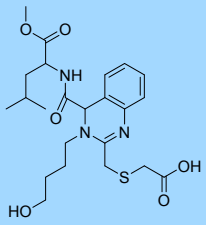
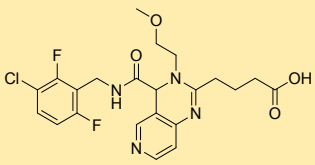
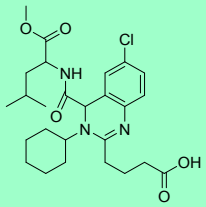
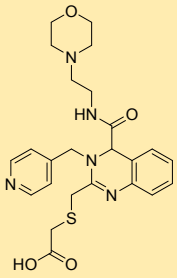
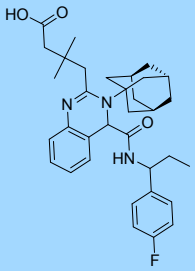
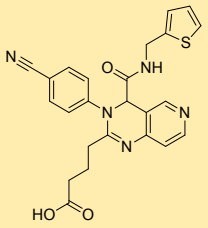
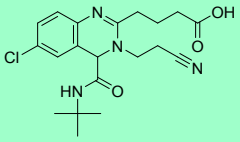
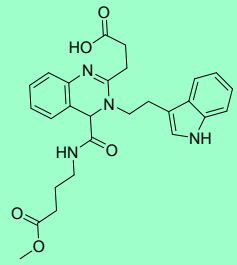
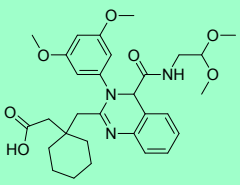
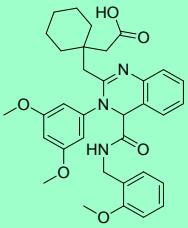
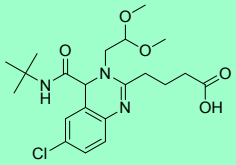
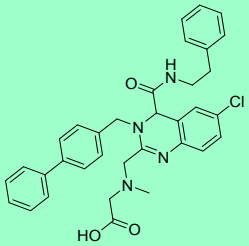
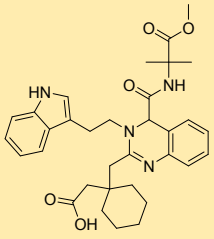
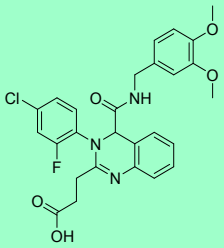
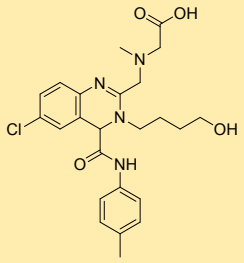
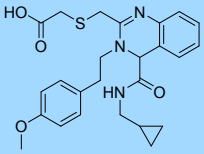
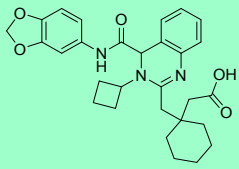
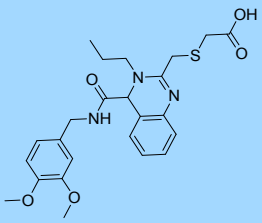
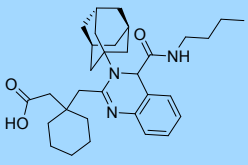
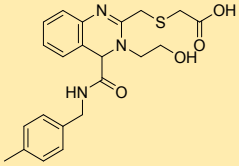
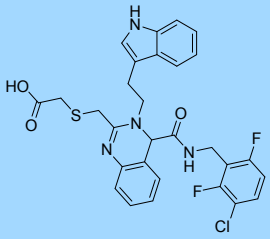
13				
14				
15				
16				
17				
18				

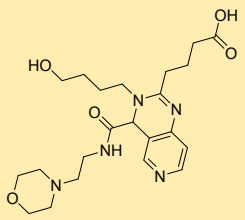
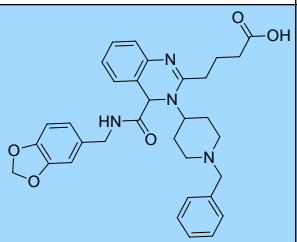
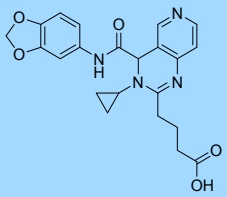
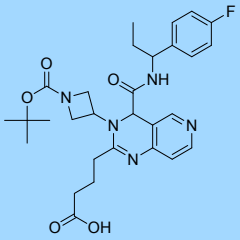
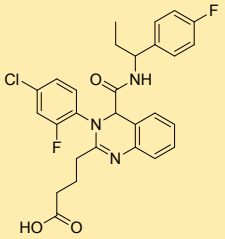
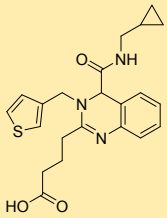
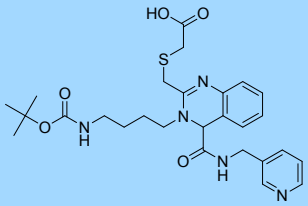
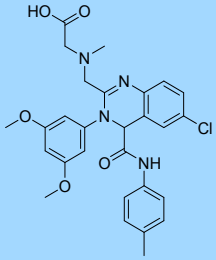
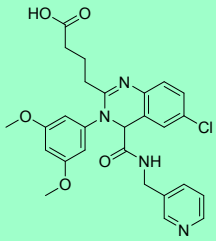
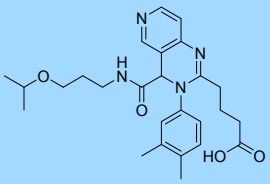
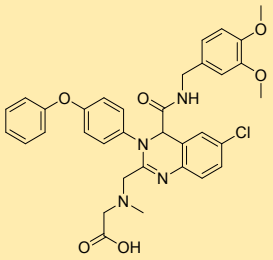
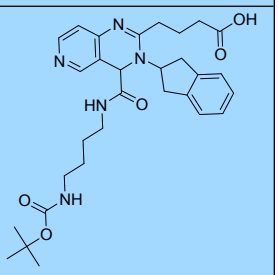
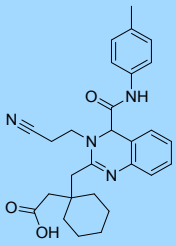
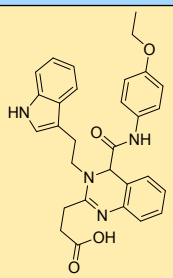
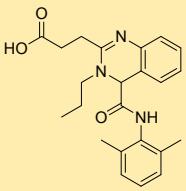
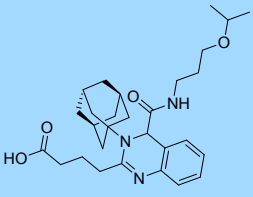
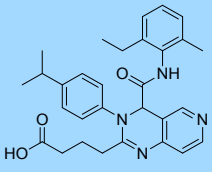
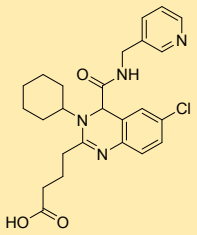
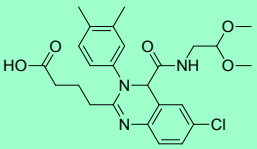
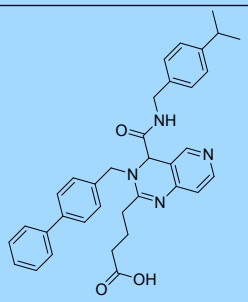
19				
20				
21				
22				
23				
24				

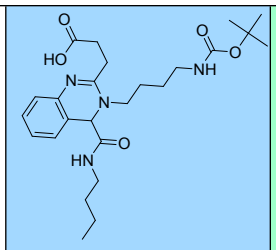
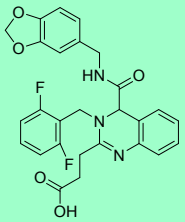
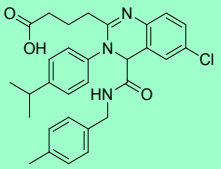
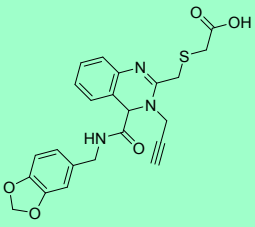
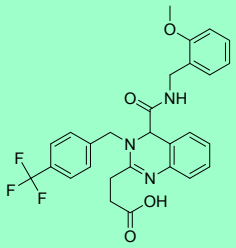
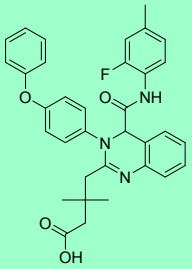
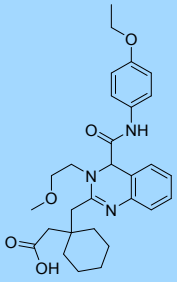
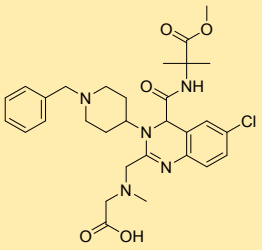
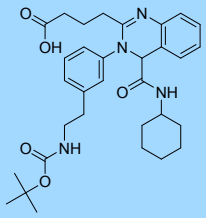
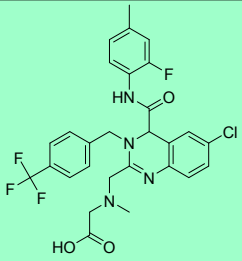
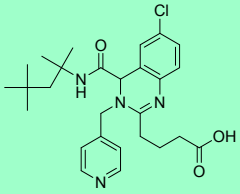
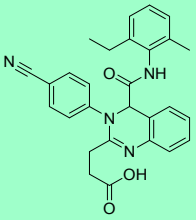
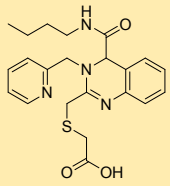
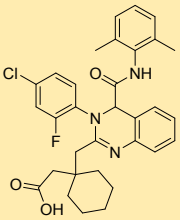
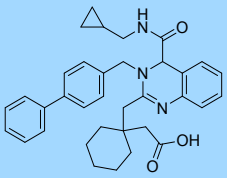
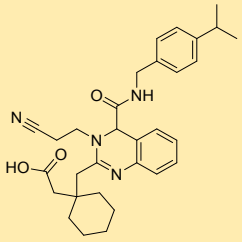
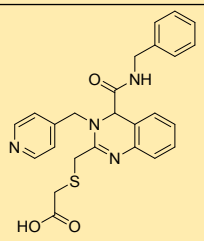
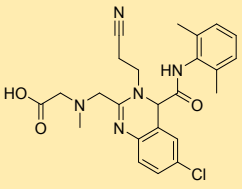
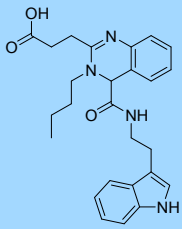
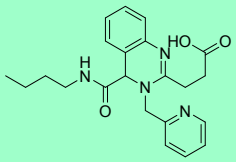
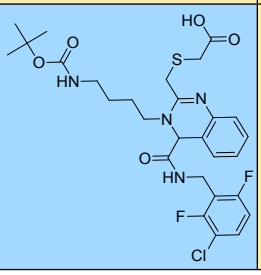
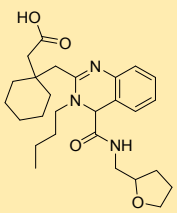
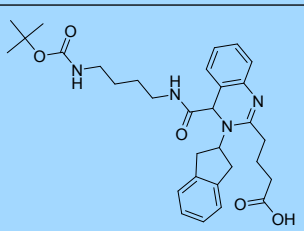
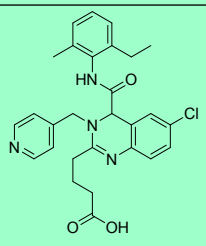
	E	F	G	H
1				
2				
3				
4				
5				
6				

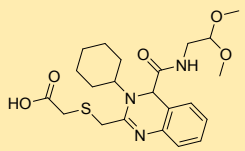
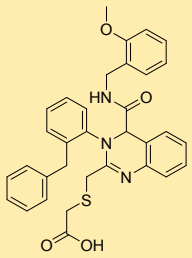
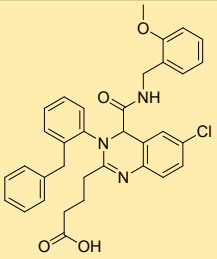
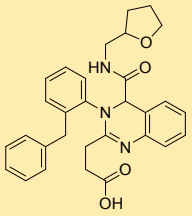
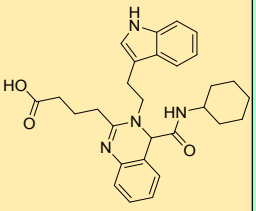
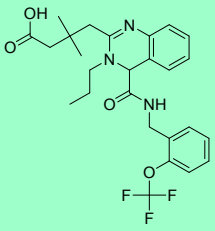
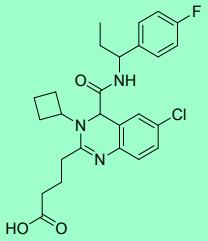
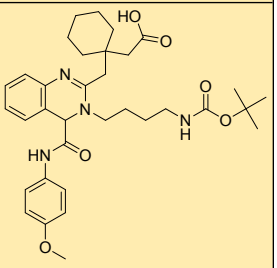
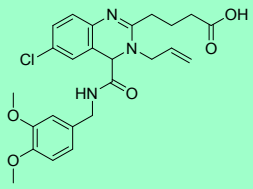
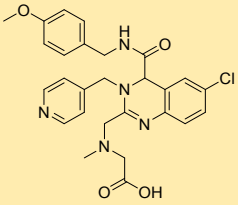
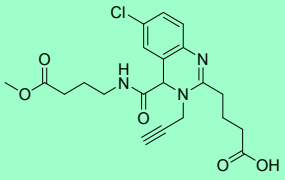
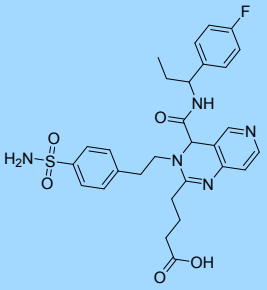
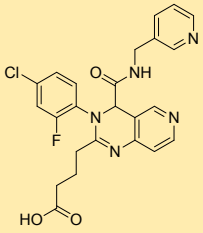
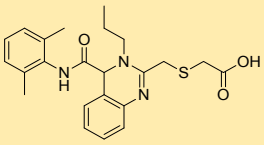
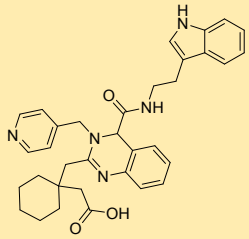
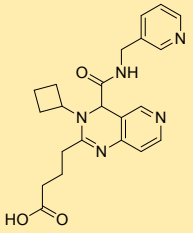
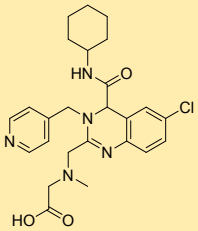
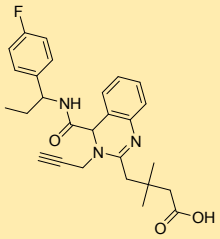
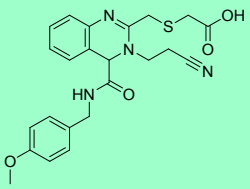
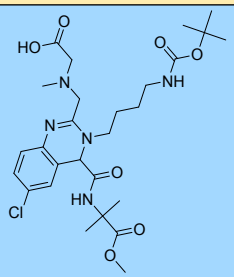
7				
8				
9				
10				
11				
12				

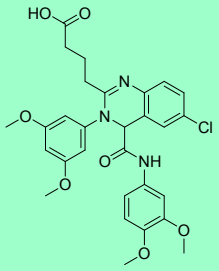
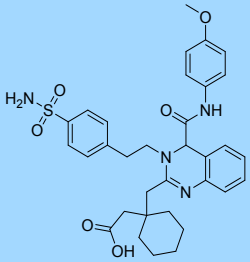
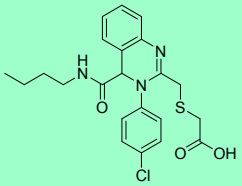
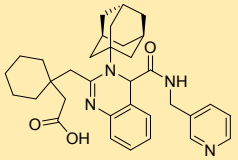
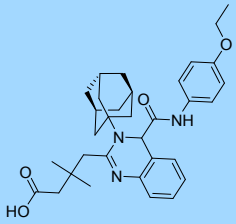
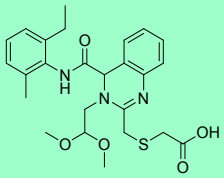
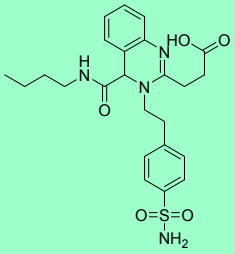
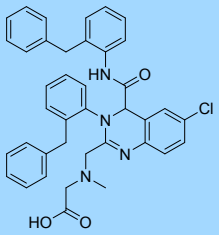
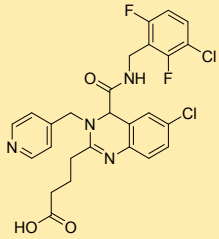
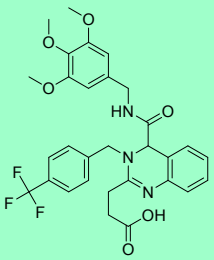
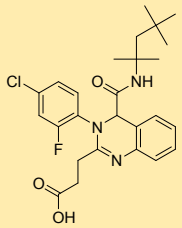
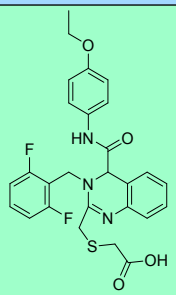
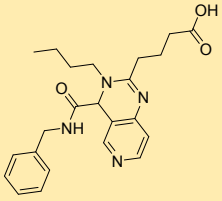
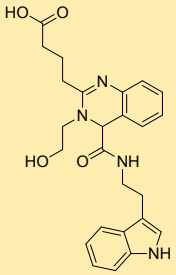
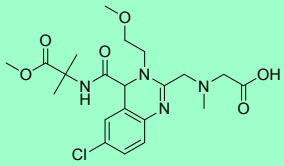
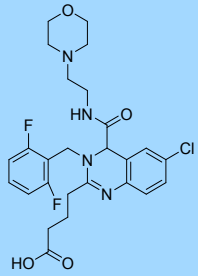
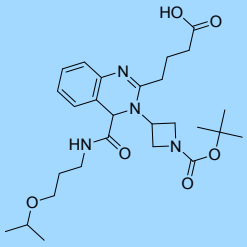
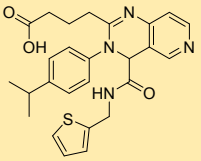
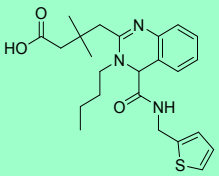
13				
14				
15				
16				
17				
18				

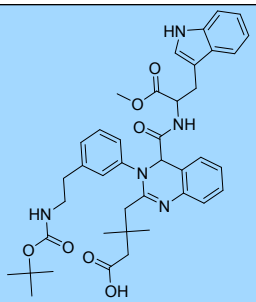
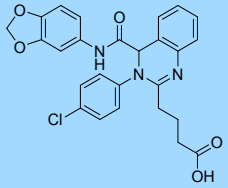
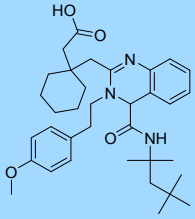
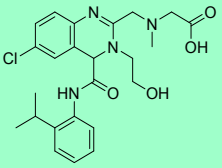
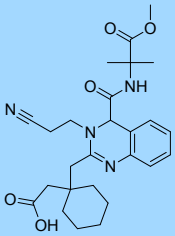
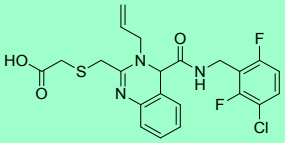
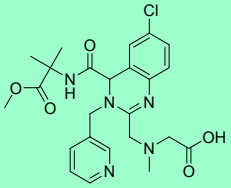
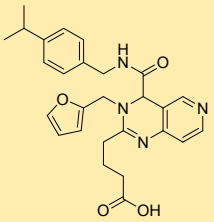
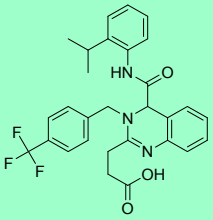
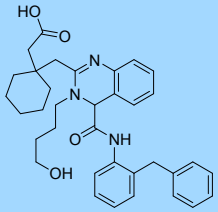
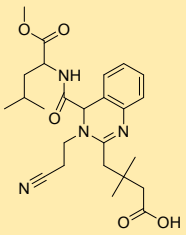
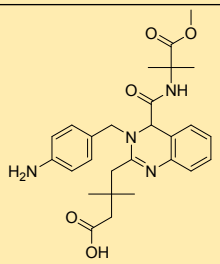
19				
20				
21				
22				
23				
24				

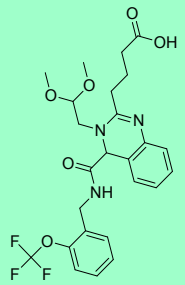
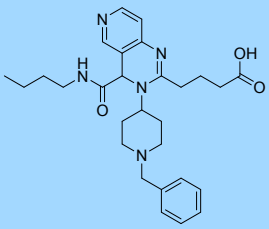
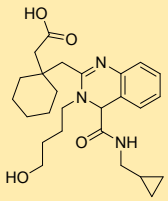
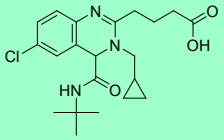
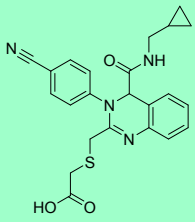
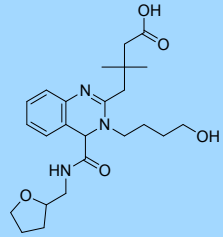
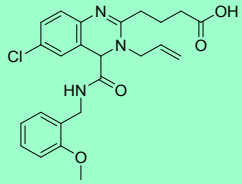
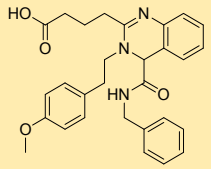
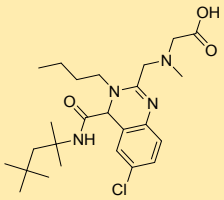
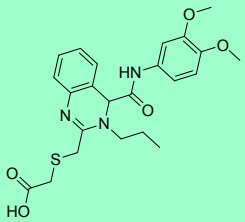
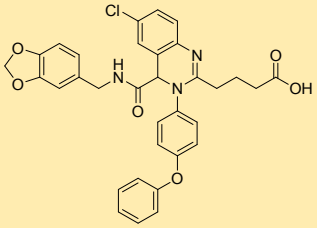
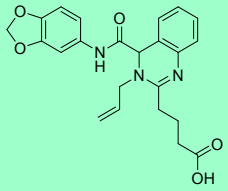
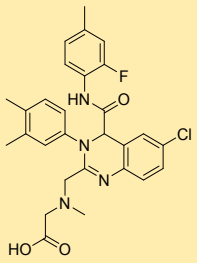
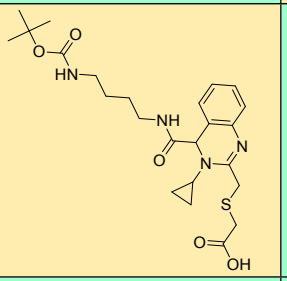
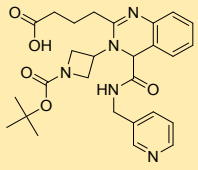
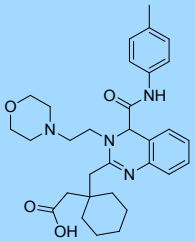
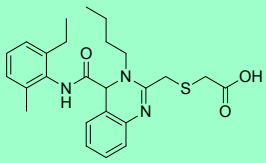
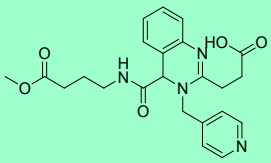
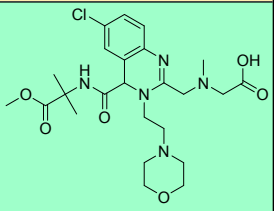
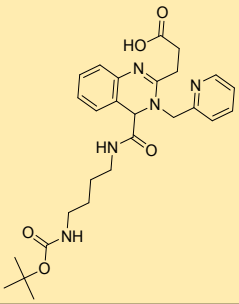
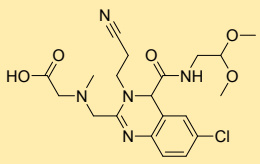
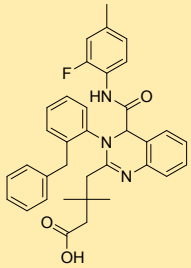
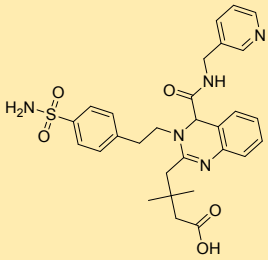
	I	J	K	L
1				
2				
3				
4				
5				

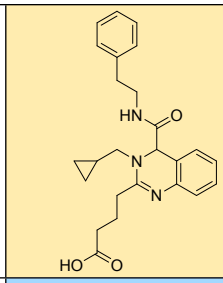
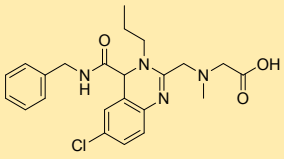
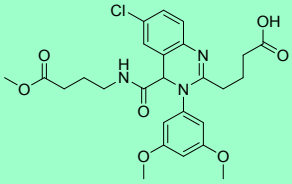
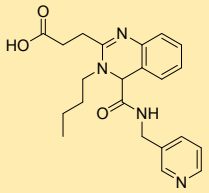
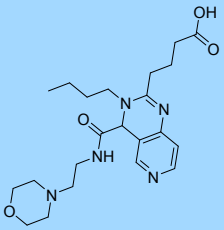
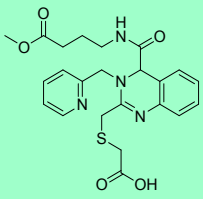
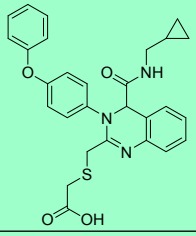
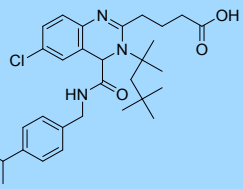
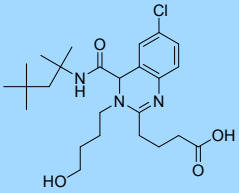
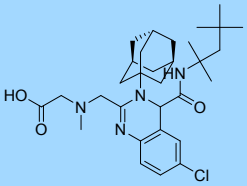
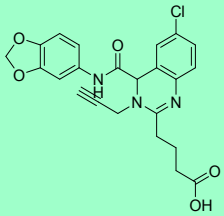
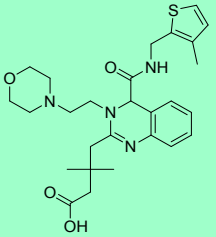
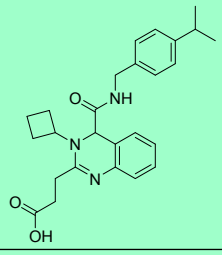
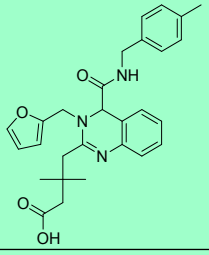
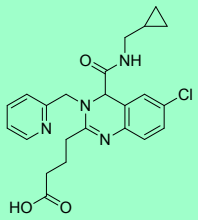
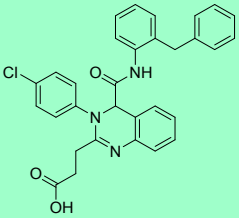
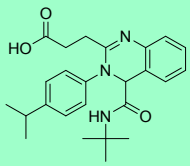
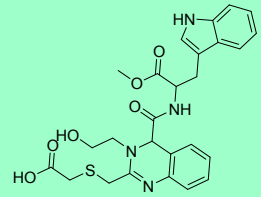
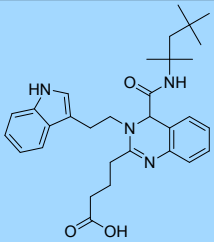
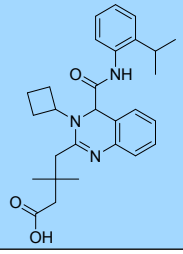
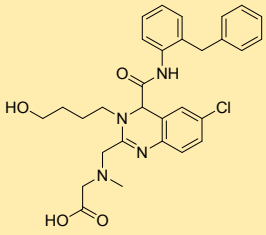
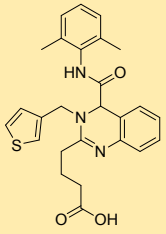
6				
7				
8				
9				
10				
11				

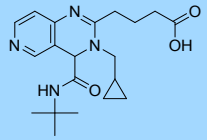
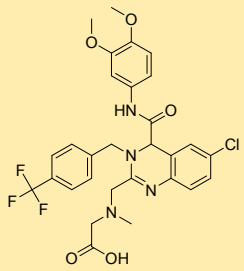
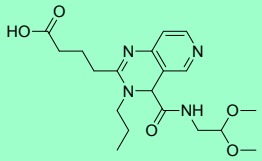
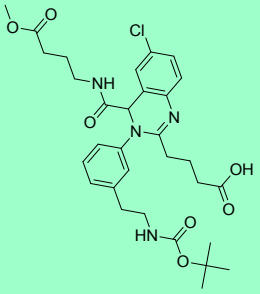
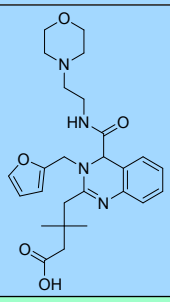
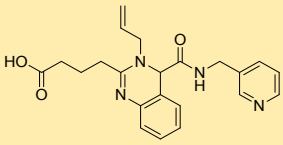
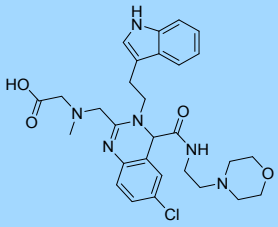
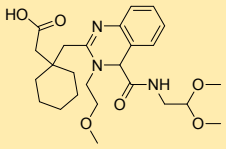
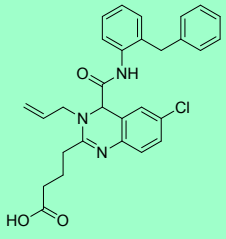
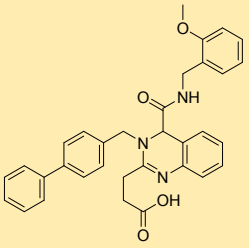
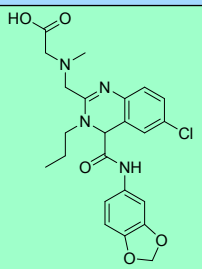
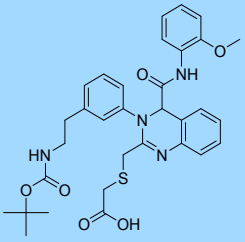
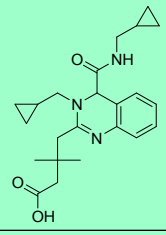
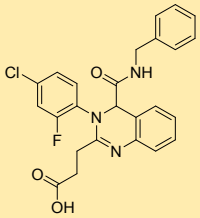
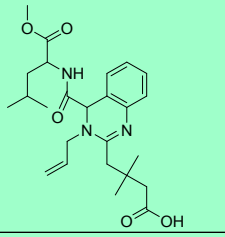
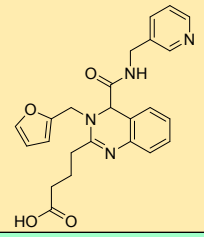
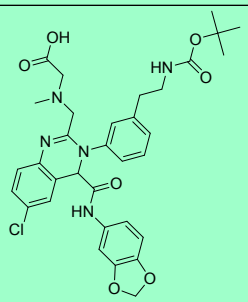
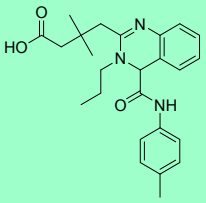
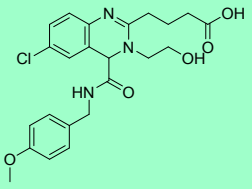
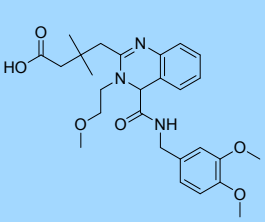
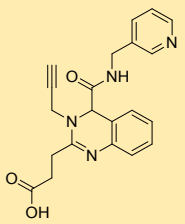
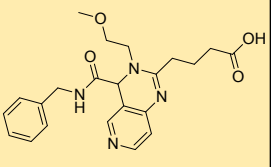
12				
13				
14				
15				
16				

17				
18				
19				
20				
21				

22				
23				
24				

	M	N	O	P
1				
2				
3				
4				
5				
6				

7				
8				
9				
10				
11				
12				

13				
14				
15				
16				
17				
18				

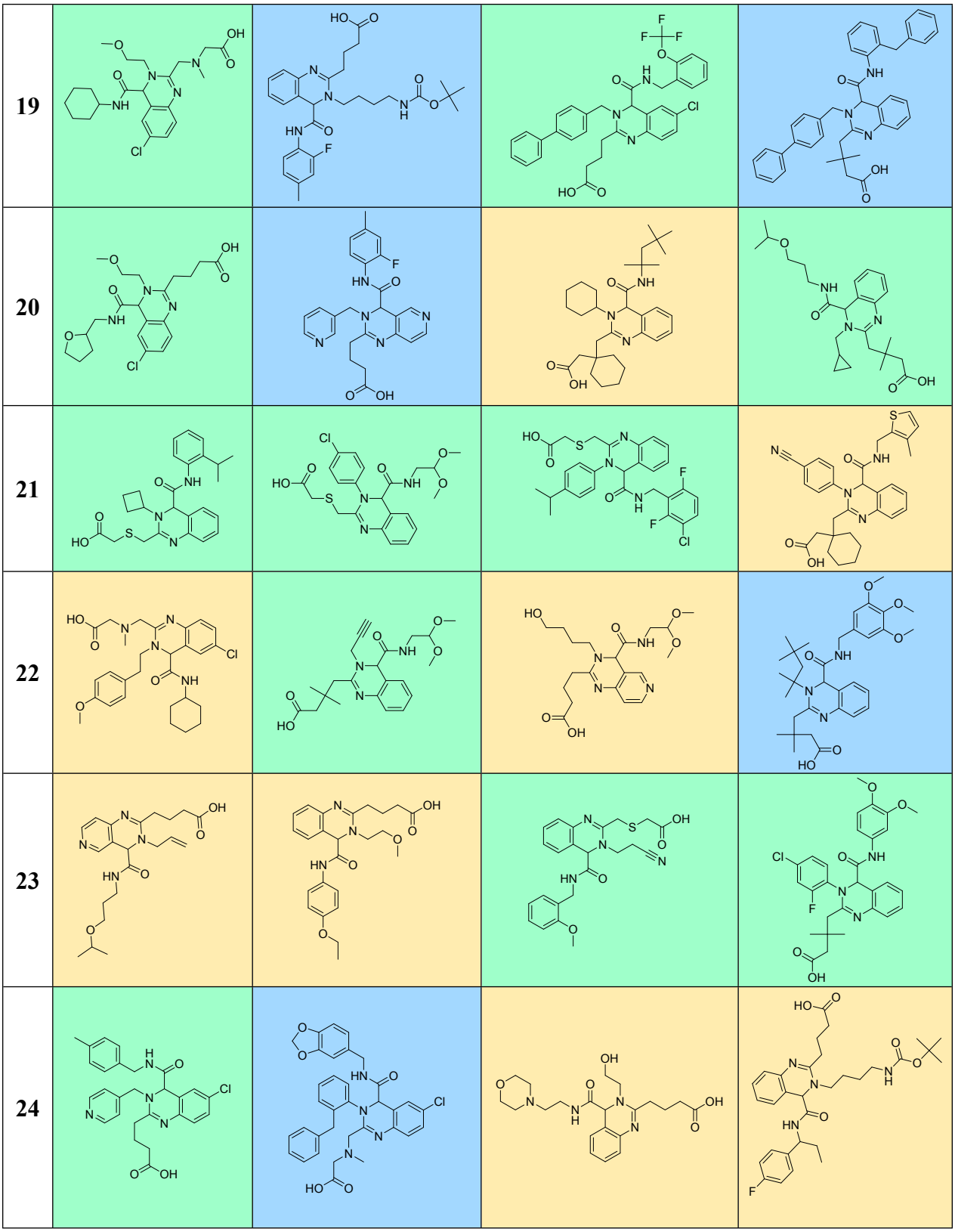


Fig. S2. Heat plots with product structures, green for major product formation, yellow for medium product formation and blue for no product formation.

3.6. Quality control (QC)

The analytics of all wells were performed by SFC-UV-MS. Mass spectra were measured on a Waters Investigator Supercritical Fluid Chromatograph with a 3100 MS Detector (ESI^+) via flow injection analysis (FIA) and MassLynx software.

Conditions: eluent composition: MeOH, 2% H_2O, 0.1% formic acid; run time: 2 min; flow rate: 1 mL/min.

Each well of the destination plate was diluted with 30 µL ethylene glycol and then the chromatographic analysis was done by SFC-MS using an autosampler.

The SFC analytic of one well took ~2 min, resulting in an overall measuring time for the 384 wells of around 13 h.

3.7. Automated analysis of mass spectrometry data

3.7.1. Preparation

Mass spectrometry (MS) data were automatically analyzed using in-house written Python software. This software makes use of the mzXML file format that first needs to be created. For this purpose, mzXML files were converted from Waters RAW using the MSConvert tool (version 3.0) from the ProteoWizard project using the default settings. These resulting files were consecutively used in the in-house written program, which is documented in an online repository (https://bitbucket.org/ca_warmerdam/auto-ms-analysis). In addition to the mass spectrometry files, a txt file with the smiles and the location of the expected product per well on the plate were entered. These locations are used both to create an output matrix and as an identifier for matching the expected product to an mzXML file. The expected time range for the bulk of the peaks was set to 0.1-1 min. The expected M/Z range that was specified as 200-800 corresponds to the range the mass spectrometer was set to detect. For running the software Python 3.6.6 was used with the additional packages which were in concordance with the requirements specified within the repository.

3.7.2. Processing mzXML files into spectra

Within the python software, mzXML files are first parsed into a queryable data structure using the Python XML parser module. Next, the MS data, comprised of scans that together represent the spectrum, are filtered in order to remove uninformative scans that are labelled with an msLevel of

0. In addition, scans that are outside of the specified time range are discarded. The contents of the scans are subsequently decoded as these are Base64 encoded by default, and the resulting values are thereafter decompressed using the decompression functionality within the zlib Python module. This results in a regular collection of pairs consisting of a mass-to-charge ratio with corresponding intensities. Intensity values were considered erroneous, and are thus removed, if they expand further than 5000 times the inter quantile range of the intensities within the specific well. Afterwards, the mass-to-charge ratios are collected in bins of size 1 around an integer value. Within this process, the intensity values are summed for every bin.

3.7.3. Prediction procedure

To assign a prediction of abundance for a product, from the peaks corresponding to all user-specified adduct masses (M+H, M+Na, M+K peaks) (± 0.5), the highest peak within a well is isolated. The intensity of this peak relative to the highest peak represents the initial prediction for a product. After having run the software, predictions with values more than 0.9 were classified as green, predictions with values less than 0.1 were classified as blue, and predictions with values equal and between these thresholds were classified as yellow.

3.7.4. Python code

Python file analyse_mass_spectrometry_data.py:

```
#!/usr/bin/env python3

"""
Script for classifying mass spectrometry mzXML files.

usage: analyse_mass_spectrometry_data.py [-h] -d DATA [DATA ...] -o
                                         OUTPUT_FILE -i
                                         SAMPLE_INFORMATION_FILE_PATH
                                         [-R RETENTION_TIME_RANGE
                                         RETENTION_TIME_RANGE]
                                         [-M MZ_RANGE MZ_RANGE]

Automatic mass spectrometry analyzer

optional arguments:
  -h, --help            show this help message and exit
  -d DATA [DATA ...], --data DATA [DATA ...]
                        Give a path for the directory where data is
                        stored to analyse. This can be multiple files.
                        The asterisk can be used as wildcard.
                        The argument 'd:\files\*.mzxml' will result
                        in all mzxml files being analyzed located
                        inside the folder: 'd:\files'
  -o OUTPUT_FILE, --output_file OUTPUT_FILE
                        Give a file path the output files can be
```

```
                written to.
                For example: 'd:\files\heatmap.xlsx'
  -i SAMPLE_INFORMATION_FILE_PATH, --sample_information_file_path
  SAMPLE_INFORMATION_FILE_PATH
                Specify a smiles file with spaces as a
                separator that
                contains a smiles in the first column and the
                location
                on the output plate in the last column.
  -R RETENTION_TIME_RANGE RETENTION_TIME_RANGE,
  --retention_time_range RETENTION_TIME_RANGE RETENTION_TIME_RANGE
                Specify the retention time to filter on.
                default is
                between 0.2 and 0.5.
  -M MZ_RANGE MZ_RANGE, --mz_range MZ_RANGE MZ_RANGE
                Specify the mass/charge range to filter on.
                default is
                between 200 and 600.
"""

import sys
import warnings

import graphical_user_interface
import mass_spectrometry_heatmap_tool

import pandas as pd

from pybel import readstring

from argument_parser import ArgumentParser
from mass_spectrum_analyzer import MassSpectraAnalyzer

__author__ = "C.A. (Robert) Warmerdam"
__credits__ = ["Robert Warmerdam"]
__status__ = "Development"


def get_mol_weight(smiles):
    """
    Returns molecular weight given a smiles
    :param smiles: smiles molecule in str
    :return: molecular weight in float
    """
    mol = readstring("smi", smiles)
    return mol.exactmass


def get_sample_information_table(sample_information_file_path):
    """
    Function processing the sample information file path to a pandas
    table.
    The file should have a spaces as a delimiter.
    :param sample_information_file_path:
    :return: numpy array containing at least 2 columns
    The output is as follows:
    First Column: Molecule in smiles
```

```
    Last Column: Location in the plate (A1, A2, B1, B2, etc.)
    """
    # Get the maximum column length
    cols = 0
    with open(sample_information_file_path) as f:
        for row in f:
            # Count the number of delimiters
            cols = max(cols, row.count(" ") + 1)

    # Create correct amount of names
    names = ['loc_' + str(i) for i in range(cols - 2)]
    names.insert(0, 'smiles')
    names.append('location')

    sample_information_table = pd.read_table(
        sample_information_file_path, sep=' ', header=None,
        names=names)
    # Get the last non nan value for every row to make sure the
    # location column holds the last value
    sample_information_table[
        'location'] = sample_information_table.ffill(axis=1).iloc[:, -1]
    # Add mol weights
    sample_information_table[
        'mol_weight'] = sample_information_table.apply(
        lambda row: get_mol_weight(row['smiles']),
        axis=1)
    # Return table / array
    return sample_information_table


def get_label_data_table(label_data):
    """
    Function processing the label data file path to a pandas table.
    The file should have a commas as a delimiter.
    :param label_data:
    :return: numpy array containing at least 2 columns
    The output is as follows:
    First Column: Location in the plate (A1, A2, B1, B2, etc.)
    Second Column: label like 0, 0.5 or 1
    """
    # Convert file to the data frame
    label_data_table = pd.read_table(label_data, sep=',', header=None,
                                     dtype={'location': str,
                                            'label': str},
                                     names=['location', 'label'],
                                     usecols=[0, 1])

    # Strip labels from spaces
    label_data_table = label_data_table.apply(
        lambda x: x.str.strip() if x.dtype == "object" else x)

    # Return table / array
    return label_data_table


def main(argv=None):
    if argv is None:
```

```
    argv = sys.argv

  if len(argv) == 1:
    app = graphical_user_interface.GraphicalUserInterface()
    app.mainloop()
    return 0

  # Parse arguments
  parser = ArgumentParser()
  parser.add_data_argument()
  parser.add_output_file_path_prefix_argument()
  parser.add_sample_information_file_path_argument()
  parser.add_ranges_argument()
  args = parser.parse_input(argv[1:])

  # Get sample information table containing smiles, locations and
  # molecular weight
  sample_information_table = get_sample_information_table(
    args.sample_information_file_path)

  # Get samples, samples
  # with warnings.catch_warnings():
  #     warnings.filterwarnings("error")
  analyzer = MassSpectraAnalyzer(args.data, sample_information_table,
                                 args.retention_time_range,
                                 args.mz_range)
  analyzer.process_spectra()

  # Get results table
  results_table = analyzer.get_results_table()

  # Export heatmaps
  mass_spectrometry_heatmap_tool.export_heatmaps(args.output_file,
                                       results_table)
  return 0


if __name__ == "__main__":
  sys.exit(main())
```

Python file argument_parser.py:

```
#!/usr/bin/env python3

"""
Script containing the definition of an argument parser built with the
argparse module
"""

import glob
import sys
import os
import argparse

__author__ = "C.A. (Robert) Warmerdam"
```

```
__credits__ = ["Robert Warmerdam"]
__version__ = "0.0.2"
__status__ = "Development"


class ArgumentParser:
    def __init__(self):
        self.parser = self.create_argument_parser()

    def parse_input(self, argv):
        """
        Parse command line input.
        :param argv: given arguments
        :return: parsed arguments
        """
        args = self.parser.parse_args(argv)
        # Compare list sizes if there is training data
        if hasattr(args, 'training_data'):
            if len(args.training_data) != len(
                    args.sample_information_file_paths):
                raise argparse.ArgumentTypeError(
                    "Did not receive the correct amount of sample "
                    "information file paths. "
                    "Should be equal to training data argument count")
            if len(args.training_data) != len(args.label_data):
                raise argparse.ArgumentTypeError(
                    "Did not receive the correct amount of label data "
                    "file paths. "
                    "Should be equal to training data argument count")

        # Flatten list if there is training data
        if hasattr(args, 'data'):
            args.data = [item for sublist in args.data for item in
                         sublist]

        # Only pass the name of the output file prefix
        if hasattr(args, 'output_file_path_prefix'):
            args.output_file_path_prefix = \
                args.output_file_path_prefix.name

        # Only pass the name of the output model
        if hasattr(args, 'model_path'):
            args.model_path = args.model_path.name

        # Check retention time range
        if hasattr(args, 'retention_time_range') and \
                args.retention_time_range[0] >= \
                args.retention_time_range[1]:
            raise argparse.ArgumentTypeError(
                "The first value should be lower than the last value")

        # Check mass-charge range
        if hasattr(args, 'mz_range') and args.mz_range[0] >= \
                args.mz_range[1]:
            raise argparse.ArgumentTypeError(
                "The first value should be lower than the last value")
```

```
    return args

def create_argument_parser(self):
    """
    Method creating an argument parser
    :return: parser
    """
    parser = argparse.ArgumentParser(
        description="Automatic mass spectrometry analyzer",
        formatter_class=argparse.RawDescriptionHelpFormatter)
    return parser

def add_sample_information_file_path_argument(self):
    self.parser.add_argument('-i', '--sample_information_file_path',
                             type=self.is_readable_file,
                             required=True,
                             default=None,
                             help="Specify a smiles file with "
                                  "spaces as a separator that "
                                  "contains a smiles in the "
                                  "first column and the location "
                                  "on the output plate in the "
                                  "last column.")

def add_data_argument(self):
    self.parser.add_argument('-d', '--data', type=self.is_data,
                             required=True, nargs='+', default=None,
                             help="Give a path for the directory "
                                  "where data is stored to "
                                  "analyse. "
                                  "{0}This can be multiple files. "
                                  "The asterisk can be used as "
                                  "wildcard. "
                                  "{0}\tThe argument "
                                  "'d:\\files\\*.mzxml'"
                                  "{0}\twill result in all mzxml "
                                  "files being analyzed located "
                                  "inside the folder:"
                                  "{0}\t'd:\\files'".format(
                                 os.linesep))

def add_label_data_argument(self):
    self.parser.add_argument('-l', '--label_data',
                             type=argparse.FileType('r'),
                             required=True, default=None,
                             help="Give a file with locations in "
                                  "the first column "
                                  "and labels (0, 0.5, 1) in the "
                                  "second column.")

def add_output_file_path_prefix_argument(self):
    self.parser.add_argument('-o', '--output_file',
                             type=argparse.FileType('wb'),
                             required=True, default=None,
                             help="Give a file path the output "
                                  "files can be written to. "
                                  "For example: "
```

```
                        "'d:\\files\\heatmap.xlsx'")

def add_ranges_argument(self):
    self.parser.add_argument('-R', '--retention_time_range',
                             type=self.is_positive_float,
                             required=False,
                             default=[0.2, 0.5],
                             nargs=2,
                             help="Specify the retention time to "
                                  "filter on. default is between "
                                  "0.2 and 0.5.")
    self.parser.add_argument('-M', '--mz_range',
                             type=self.is_positive_float,
                             required=False,
                             default=[200.0, 600.0],
                             nargs=2,
                             help="Specify the mass/charge range "
                                  "to filter on. default is "
                                  "between 200 and 600.")

@staticmethod
def is_positive_float(value):
    """
    Checks whether a
    :param value: the string to check.
    :return: a checked float value.
    """
    float_value = float(value)
    if float_value <= 0:
        raise argparse.ArgumentTypeError(
            "{} is an invalid float value (must be above "
            "0.0)".format(
                value))
    return float_value

@staticmethod
def is_readable_dir(train_directory):
    """
    Checks whether the given directory is readable
    :param train_directory: a path to a directory in string format
    :return: train_directory
    :raises: Exception: if the given path is invalid
    :raises: Exception: if the given directory is not accessible
    """
    if not os.path.isdir(train_directory):
        raise Exception("directory: {0} is not a valid path".format(
            train_directory))
    if os.access(train_directory, os.R_OK):
        return train_directory
    else:
        raise Exception(
            "directory: {0} is not a readable dir".format(
                train_directory))

@staticmethod
def is_readable_file(file_path):
    """
```

```
        Checks whether the given directory is readable
        :param file_path: a path to a file in string format
        :return: file_path
        :raises: Exception: if the given path is invalid
        :raises: Exception: if the given directory is not accessible
        """
        if not os.path.isfile(file_path):
            raise Exception(
                "file path:{0} is not a valid file path".format(
                    file_path))
        if os.access(file_path, os.R_OK):
            return file_path
        else:
            raise Exception(
                "file path:{0} is not a readable file".format(
                    file_path))

    @staticmethod
    def is_data(argument):
        """
        Parses a string that specifies the path(s) of the data.
        :param argument:
        :return:
        """
        # If an asterisk is the string, expand using glob.
        if '*' in argument:
            paths = glob.glob(argument)
            files = []
            for file_path in paths:
                files.append(argparse.FileType('r')(file_path))
            if len(files) == 0:
                raise argparse.ArgumentTypeError(
                    argument + ' matches exactly zero files.')
            return files
        else:
            # Use default argparse filename parser if there is not an
            # asterisk in the string.
            return argparse.FileType('r')(argument)

    def is_writable_location(self, path):
        self.is_readable_dir(os.path.dirname(path))
        return path


def main(argv=None):
    if argv is None:
        argv = sys.argv
    return 0


if __name__ == "__main__":
    sys.exit(main())
```

Python file graphical_user_interface.py:

```
#!/usr/bin/env python3
"""
Script responsible for serving a graphical user interface for
analyzing mass spectra.

usage: graphical_user_interface.py
"""

import sys
import tkinter as tk
import warnings
from argparse import ArgumentError

import analyse_mass_spectrometry_data
import mass_spectrometry_heatmap_tool

from argument_parser import ArgumentParser
from mass_spectrum_analyzer import MassSpectraAnalyzer
from sample import Sample
from tkinter import filedialog, messagebox

__author__ = "C.A. (Robert) Warmerdam"
__credits__ = ["Robert Warmerdam"]
__status__ = "Development"


class GraphicalUserInterface(tk.Tk):

    def __init__(self, *args, **kwargs):
        tk.Tk.__init__(self, *args, **kwargs)
        container = tk.Frame(self)

        container.pack(side="top", fill="both", expand=True)

        container.grid_rowconfigure(0, weight=1)
        container.grid_columnconfigure(0, weight=1)

        self.frames = {}

        frame = MainPage(container, self)

        self.frames[MainPage] = frame

        frame.grid(row=0, column=0, columnspan=3, sticky="nsew", padx=4,
                   pady=4)

    def show_frame(self, cont):
        frame = self.frames[cont]
        frame.tkraise()


class MainPage(tk.Frame):

    def __init__(self, parent, controller):
        tk.Frame.__init__(self, parent)
        self.winfo_toplevel().title("Mass spectrum analyzer")
```

```
        self.grid_rowconfigure(0, weight=1)
        self.grid_columnconfigure(0, weight=1)

        self.mz_xml_file_paths = list()
        self.smiles_file_path = None
        self.output_file_path = None
        self.retention_time_range = (0, 0)
        self.mz_range = (0, 0)
        self._min_mz_range_var = tk.StringVar()
        self._min_retention_time_var = tk.StringVar()
        self._max_mz_range_var = tk.StringVar()
        self._max_retention_time_var = tk.StringVar()
        self._ion_name = tk.StringVar()
        self._ion_mass_multiplier = tk.StringVar()
        self._ion_mass_addition = tk.StringVar()

        self._mz_xml_files_entry = tk.Entry(self)
        self._smiles_file_entry = tk.Entry(self)
        self._output_file_entry = tk.Entry(self)
        self._min_mz_range_entry = tk.Entry(
            self, textvariable=self._min_mz_range_var)
        self._max_mz_range_entry = tk.Entry(
            self, textvariable=self._max_mz_range_var)
        self._min_time_range_entry = tk.Entry(
            self, textvariable=self._min_retention_time_var)
        self._max_time_range_entry = tk.Entry(
            self, textvariable=self._max_retention_time_var)
        self._adduct_mass_label_entry = tk.Entry(
            self, textvariable=self._ion_name)
        self._adduct_mass_multiplication_entry = tk.Entry(
            self, textvariable=self._ion_mass_multiplier)
        self._adduct_mass_addition_entry = tk.Entry(
            self, textvariable=self._ion_mass_addition)
        self._frequent_adduct_scroll = tk.Scrollbar(
            self, orient=tk.VERTICAL)
        self._frequent_adduct_listbox = tk.Listbox(
            self, yscrollcommand=self._frequent_adduct_scroll.set,
            height=6)

        self.adduct_entries = Sample.frequent_adduct_masses

        self.create_widgets()

    def create_widgets(self):
        """
        Create widgets
        """
        # Make everything sticky to let the flow in unused space
        # Create and place mz xml files input
        label = tk.Label(self, text="Select mzXML files")
        label.grid(row=1, column=0, columnspan=3, sticky="nsew")
        self._mz_xml_files_entry.grid(row=2, column=0, columnspan=3,
                                      sticky="nsew")
        browse_button = tk.Button(self, text="browse",
                                  command=self.browse_for_mz_xml_files)
        browse_button.grid(row=2, column=3, columnspan=2, sticky="nsew")
```

```
# Create and place smiles file input
label = tk.Label(self, text="Select smiles file")
label.grid(row=3, column=0, columnspan=3, sticky="nsew")
self._smiles_file_entry.grid(row=4, column=0, columnspan=3,
                             sticky="nsew")
browse_button = tk.Button(self, text="browse",
                          command=self.browse_for_smiles_file)
browse_button.grid(row=4, column=3, columnspan=2, sticky="nsew")

# Create and place output file input
label = tk.Label(self, text="Enter output file")
label.grid(row=5, column=0, columnspan=3, sticky="nsew")
self._output_file_entry.grid(row=6, column=0, columnspan=3,
                             sticky="nsew")
browse_button = tk.Button(self, text="browse",
                          command=self.browse_for_output_file)
browse_button.grid(row=6, column=3, columnspan=2, sticky="nsew")

# Place inputs for m/z range
label = tk.Label(self, text="M/Z range")
label.grid(row=7, column=0, columnspan=3, sticky="nsew")
# Use for sticky only w(west) to maintain constant width
self._min_mz_range_entry.grid(row=8, columnspan=2, column=0,
                              sticky="w")
label = tk.Label(self, text="-")
label.grid(row=8, column=1, sticky="nsew")
self._max_mz_range_entry.grid(row=8, column=2, sticky="nsew")
# Insert default values
self._min_mz_range_entry.insert(1, 200)
self._max_mz_range_entry.insert(1, 600)

# Place inputs for time range
label = tk.Label(self, text="Time range")
label.grid(row=9, column=0, columnspan=3, sticky="nsew")
# Use for sticky only w(west) to maintain constant width
self._min_time_range_entry.grid(row=10, column=0, sticky="w")
label = tk.Label(self, text="-")
label.grid(row=10, column=1, sticky="nsew")
self._max_time_range_entry.grid(row=10, column=2, sticky="e")
# Place default values
self._min_time_range_entry.insert(1, 0.2)
self._max_time_range_entry.insert(1, 0.5)

# Place inputs for adduct entries
label = tk.Label(self, text="frequent adduct masses")
label.grid(row=11, column=0, columnspan=3, sticky="nsew")
label = tk.Label(self, text="Name")
label.grid(row=12, column=0, columnspan=2, sticky="w")
self._adduct_mass_label_entry.grid(row=12, column=2,
                                   columnspan=1, sticky="nsew")
label = tk.Label(self, text="Multiplication factor")
label.grid(row=13, column=0, columnspan=2, sticky="w")
self._adduct_mass_multiplication_entry.grid(row=13, column=2,
                                            columnspan=1,
                                            sticky="nsew")
label = tk.Label(self, text="Addition of mass")
label.grid(row=14, column=0, columnspan=2, sticky="w")
```

```
        self._adduct_mass_addition_entry.grid(row=14, column=2,
                                              columnspan=1,
                                              sticky="nsew")

        # Place buttons for adding and removing adduct entries
        addition_button = tk.Button(self, text="Add",
                                    command=self.add_adduct_entry)
        addition_button.grid(row=15, column=0, sticky="nsew")
        delete_button = tk.Button(self, text="Delete",
                                  command=self.remove_adduct_entry)
        delete_button.grid(row=15, column=2, sticky="nsew")

        # Place list of adduct entries
        self._frequent_adduct_scroll.config(
            command=self._frequent_adduct_listbox.yview)
        self._frequent_adduct_scroll.grid(row=16, column=3,
                                          sticky="nsw")
        self._frequent_adduct_listbox.grid(row=16, column=0,
                                           columnspan=3, sticky="nsew")

        # Update adduct entries
        self.update_frequent_adduct_listbox()

        # Place submit button
        submit_button = tk.Button(self, text="Run", command=self.run)
        submit_button.grid(row=17, column=0, columnspan=3,
                           sticky="nsew")

    def run(self):
        """
        Run mass spectrum analysis
        """
        try:
            # Set mz range
            self.set_mz_range()

            # Set retention time range
            self.set_retention_time_range()

            # Get mz data
            data = self.get_mz_data()

            # Get sample information table containing smiles,
            # locations and molecular weight
            sample_information_table = self.get_smiles_data()

            # Get samples, samples
            with warnings.catch_warnings():
                warnings.filterwarnings("error")
                analyzer = MassSpectraAnalyzer(data,
                                               sample_information_table,
                                               self.retention_time_range,
                                               self.mz_range)
                analyzer.process_spectra()

            # Get results table
            results_table = analyzer.get_results_table()
```

```
            # Export heatmaps
            mass_spectrometry_heatmap_tool.export_heatmaps(
                self.output_file_path, results_table)
        except (ValueError, ArgumentError) as e:
            messagebox.showinfo("Error", "{}.".format(str(e)))

    def get_smiles_data(self):
        """
        Get smiles data table
        :return: Smiles data table, ValueError if not entered.
        """
        if len(self.smiles_file_path) < 1:
            raise ValueError("no smiles file entered")

        sample_information_table = \
            analyse_mass_spectrometry_data.get_sample_information_table(
            self.smiles_file_path)
        return sample_information_table

    def get_mz_data(self):
        """
        Get mz data
        :return: Files, ValueError if not entered.
        """
        # Check length
        if len(self.mz_xml_file_paths) < 1:
            raise ValueError("no mzXML files entered")
        # Open data files
        data = [ArgumentParser.is_data(path) for path in
                self.mz_xml_file_paths]
        return data

    def update_frequent_adduct_listbox(self):
        """
        Updates adduct entries listbox.
        """
        self._frequent_adduct_listbox.delete(0, tk.END)
        for entry in self.adduct_entries:
            self._frequent_adduct_listbox.insert(tk.END, "; ".join(
                [str(value) for value in entry]))

    def browse_for_mz_xml_files(self):
        """
        Browses for multiple files and sets mz xml files entry
        """
        self.mz_xml_file_paths = list(filedialog.askopenfilenames())
        self._mz_xml_files_entry.delete(0, tk.END)
        self._mz_xml_files_entry.insert(1, ";".join(
            self.mz_xml_file_paths))

    def browse_for_smiles_file(self):
        """
        Browses for a file and sets smiles file entry
        """
        self.smiles_file_path = filedialog.askopenfilename()
        self._smiles_file_entry.insert(1, self.smiles_file_path)
```

```
def browse_for_output_file(self):
    """
    Browses directories for a new file (or for overwriting a file)
    """
    self.output_file_path = filedialog.asksaveasfilename()
    # If the excel extension is not used, add id to the filename
    if self.output_file_path.endswith(".xlsx") is False:
        self.output_file_path += ".xlsx"
    self._output_file_entry.insert(1, self.output_file_path)

def add_adduct_entry(self):
    """
    Adds adduct entries
    :return: 1 if ValueErrors were raised while an adduct entry.
    """
    # Get values
    ion_name_value = self._ion_name.get()
    ion_mass_multiplier_value = self._ion_mass_multiplier.get()
    ion_mass_addition_value = self._ion_mass_addition.get()

    try:
        # Check if ion name has no length
        if len(ion_name_value) == 0:
            # Alert user
            raise ValueError("Ion name not entered")

        # Check if multiplier can be converted to a float
        try:
            ion_mass_multiplier_float_value = self.get_value(
                ion_mass_multiplier_value)
        except ValueError as e:
            # Alert user
            raise ValueError(
                "Ion mass multiplier {}".format(str(e)))

        # Check if addition value length and if it can be
        # converted to a float
        try:
            ion_mass_addition_float_value = self.get_value(
                ion_mass_addition_value)
        except ValueError as e:
            # Alert user
            raise ValueError("Ion mass addition {}".format(str(e)))
    except ValueError as e:
        messagebox.showinfo("Error", "{}.".format(str(e)))
        return 1

    self.adduct_entries.append((ion_name_value,
                                ion_mass_multiplier_float_value,
                                ion_mass_addition_float_value))
    self.update_frequent_adduct_listbox()

def remove_adduct_entry(self):
    """
    Removes an adduct entry at the selected index of the adduct
    entry listbox
```

```
        """
        selected_items = self._frequent_adduct_listbox.curselection()
        for index in selected_items:
            del self.adduct_entries[index]
        self.update_frequent_adduct_listbox()

    def set_mz_range(self):
        """
        Sets the M/Z range from the min and max entries
        """
        # Get min time value
        try:
            min_mz_range_float_value = self.get_value(
                self._min_mz_range_var.get())
        except ValueError as e:
            raise ValueError("M/Z range minimum {}".format(str(e)))

        # Get max time value
        try:
            max_mz_range_float_value = self.get_value(
                self._max_mz_range_var.get())
        except ValueError as e:
            raise ValueError("M/Z range maximum {}".format(str(e)))

        # Set value
        self.mz_range = (
        min_mz_range_float_value, max_mz_range_float_value)

    def set_retention_time_range(self):
        """
        Sets the retention time range from the min and max entries
        """
        # Get min time value
        try:
            min_time_range_float_value = self.get_value(
                self._min_retention_time_var.get())
        except ValueError as e:
            raise ValueError(
                "Retention time range minimum {}".format(str(e)))

        # Get max time value
        try:
            max_time_range_float_value = self.get_value(
                self._max_retention_time_var.get())
        except ValueError as e:
            raise ValueError(
                "Retention time range maximum {}".format(str(e)))

        self.retention_time_range = (
        min_time_range_float_value, max_time_range_float_value)

    @staticmethod
    def get_value(value):
        """
        Converts value to float if it has length.
        Raises ValueError if either the length was not more than 0 or
        if the value could not be converted to float.
```

```
        :param value: Value to convert to float
        :return: The converted value
        """
        # Check if min mz range has no length
        if len(value) == 0:
            # Alert user
            raise ValueError("not entered")

        # Check if min mz range can be converted to a float
        try:
            return float(value)
        except ValueError as e:
            raise ValueError(
                "is not a numeric value ('{}')".format(value))


def main(argv=None):
    if argv is None:
        argv = sys.argv
    app = GraphicalUserInterface()
    app.mainloop()
    return 0


if __name__ == "__main__":
    sys.exit(main())
```

Python file mass_spectrometry_heatmap_tool.py:

```
#!/usr/bin/env python3
"""
usage: python mass_spectrometry_heatmap_tool.py [-h] -l LABEL_DATA
                                    -o OUTPUT_FILE -i
                                    SAMPLE_INFORMATION_FILE_PATH

Automatic mass spectrometry analyzer

optional arguments:
  -h, --help            show this help message and exit
  -l LABEL_DATA, --label_data LABEL_DATA
                        Give a file with locations in the first
                        column and
                        labels (0, 0.5, 1) in the second column.
  -o OUTPUT_FILE, --output_file OUTPUT_FILE
                        Give a output file path the output file can
                        be written
                        to. For example: 'd:\files\heatmap.xlsx'
  -i SAMPLE_INFORMATION_FILE_PATH, --sample_information_file_path
  SAMPLE_INFORMATION_FILE_PATH
                        Specify a smiles file with spaces as a
                        separator that
                        contains a smiles in the first column and the
                        location
                        on the output plate in the last column.
"""
```

```
import re
import sys
import analyse_mass_spectrometry_data

import xlsxwriter as xlsxwriter

from xlsxwriter.utility import xl_rowcol_to_cell
from argument_parser import ArgumentParser

__author__ = "C.A. (Robert) Warmerdam"
__credits__ = ["Robert Warmerdam"]
__status__ = "Development"


def add_compact_heatmap_worksheet(sample_information_table, workbook):
    """
    Adds a worksheet with a compact heatmap
    :param sample_information_table:
    :param workbook:
    """
    # Add the worksheet
    worksheet = workbook.add_worksheet('compact-heatmap')

    # Start from the first cell. Rows and columns are zero indexed.
    row = 0
    col = 0

    # Define format for prediction cells
    cell_format = workbook.add_format()
    cell_format.set_align('center')

    # Iterate through rows of information table
    letters = set()
    numbers = set()
    for index, sample_row in sample_information_table.iterrows():
        loc = sample_row['location']
        prediction = sample_row['prediction']

        # Get letters and numbers from identifier
        match = re.match(r"([A-Z]+)([0-9]+)", loc, re.I)

        if match:
            letter = ord(match.groups()[0]) - 64
            letters.add(letter)
            number = int(match.groups()[1])
            numbers.add(number)
            worksheet.write(letter, number,
                            round(prediction, 1), cell_format)

    # Set header cell format
    header_cell_format = workbook.add_format()
    header_cell_format.set_align('center')
    header_cell_format.set_bold()

    # Write column and row names
    worksheet.write_row(row, col + 1, list(range(1, max(numbers) + 1)),
```

```
                header_cell_format)
    worksheet.write_column(row + 1, col, [chr(i + 64) for i in
                                range(1, max(letters) + 1)],
                    header_cell_format)

    # Add color scale
    cell_ranges = '{}:{}'.format(xl_rowcol_to_cell(1, 1),
                            xl_rowcol_to_cell(max(letters),
                                              max(numbers)))
    worksheet.conditional_format(cell_ranges,
                            {'type': '3_color_scale'})

    # Set column widths
    worksheet.set_column(0, max(numbers), 3.57)
    return 0


def add_smiles_heatmap_worksheet(sample_information_table, workbook):
    # Add the worksheet
    worksheet = workbook.add_worksheet('smiles-heatmap')

    # Set cell size
    worksheet.set_default_row(41)
    worksheet.set_row(0, 20)

    # Start from the first cell. Rows and columns are zero indexed.
    row = 0
    col = 0

    # Iterate through rows of information table
    letters = set()
    numbers = set()

    # Iterate through rows of information table
    for index, sample_row in sample_information_table.iterrows():
        loc = sample_row['location']
        smiles_structure = sample_row['smiles']
        prediction = sample_row['prediction']

        # Get letters and numbers from identifier
        match = re.match(r"([A-Z]+)([0-9]+)", loc, re.I)

        if match:
            letter = ord(match.groups()[0]) - 64
            letters.add(letter)
            number = int(match.groups()[1])
            numbers.add(number)
            worksheet.write(number, letter, smiles_structure)
            worksheet.write(number, letter + 26, prediction)

    # Set header cell format
    header_cell_format = workbook.add_format()
    header_cell_format.set_align('center')

    # Write column and row names
    worksheet.write_column(row + 1, col,
                    list(range(1, max(numbers) + 1)),
```

```
                    header_cell_format)
    worksheet.write_row(row, col + 1, [chr(i + 64) for i in
                                range(1, max(letters) + 1)],
                        header_cell_format)

    # Add color scale
    # 1 to indicate the second row and col, 26 to shift it with the
    # length of the alphabet across columns
    cell_ranges = '{}:{}'.format(xl_rowcol_to_cell(1, 1 + 26),
                                 xl_rowcol_to_cell(max(numbers),
                                                   max(letters) + 26))
    worksheet.conditional_format(cell_ranges,
                                 {'type': '3_color_scale'})

    # Set cell size
    worksheet.set_column(0, 0, 3.14)
    worksheet.set_column(1, max(letters), 16)

    return 0


def add_sample_information_table_worksheet(sample_information_table,
                                           workbook):
    # Add the worksheet
    worksheet = workbook.add_worksheet('table')

    # Iterate through rows of information table
    for index, sample_row in sample_information_table.iterrows():
        # Write row to the worksheet
        row_list = list(sample_row)
        worksheet.write_row(index, 0, row_list)

    return 0


def add_specific_table_worksheet(sample_information_table, worksheet,
                                 label):
    # Set cell size
    worksheet.set_default_row(112.5)

    # Start at the origin of the worksheet
    row = 0
    column = 0

    # Iterate through rows of information table
    for index, sample_row in sample_information_table.iterrows():
        loc = sample_row['location']
        smiles_structure = sample_row['smiles']
        prediction = sample_row['prediction']

        if prediction == label:
            worksheet.write(row, column, smiles_structure)

            # Set following cell
            column += 1
            if column == 4:
                column = 0
```

```
            row += 1

    # Set cell size
    worksheet.set_column(0, 4, 20)

    return 0


def update_sample_information_table(label_data_table,
                                    sample_information_table):
    """
    Extends the sample information table with label data.
    :param label_data_table: The table with sample locations and labels.
    :param sample_information_table: The table with sample locations
    and smiles strings.
    :return:
    """
    for index, row in sample_information_table.iterrows():
        label_row = list(
            filter(lambda x: x[1]['location'] == row['location'],
                   label_data_table.iterrows()))
        if len(label_row) == 1:
            label_row = label_row[0][1]
            if not label_row['label'] in ('0', '0.5', '1'):
                print("Sample '{}' does not have a valid label!".format(
                    label_row['location']), file=sys.stderr)
                continue
        else:
            print(
            "Sample '{}' should only be present a single time in the "
            "label file".format(
                row['location']),
            file=sys.stderr)
            continue
        sample_information_table.at[index, 'prediction'] = float(
            label_row['label'])
    # Filter sample information table
    sample_information_table = sample_information_table[
        sample_information_table['prediction'].isin([float(0),
                                                     float(0.5),
                                                     float(1)])]
    return sample_information_table


def export_heatmaps(output_file, sample_information_table):
    """
    Function that export heatmaps in an excel file.
    :param output_file: The excel file name
    :param sample_information_table: The table that contains
    information about locations, labels and structure.
    """
    # Create a workbook
    print("Exporting heatmaps...")
    workbook = xlsxwriter.Workbook(output_file)
    # Add the compact worksheet
    add_compact_heatmap_worksheet(sample_information_table, workbook)
    # Add the smiles heatmap
```

```
    add_smiles_heatmap_worksheet(sample_information_table, workbook)
    # Add the sample information table
    add_sample_information_table_worksheet(sample_information_table,
                                           workbook)
    # Add the green table
    green = workbook.add_worksheet('green')
    add_specific_table_worksheet(sample_information_table, green, 1)
    green = workbook.add_worksheet('yellow')
    add_specific_table_worksheet(sample_information_table, green, 0.5)
    # Close and exit
    workbook.close()
    print("Done!")


def main(argv=None):
    if argv is None:
        argv = sys.argv

    # Parse arguments
    parser = ArgumentParser()
    parser.add_label_data_argument()
    parser.add_output_file_path_prefix_argument()
    parser.add_sample_information_file_path_argument()
    args = parser.parse_input(argv[1:])

    # Get sample information table containing smiles, locations and
    # molecular weight
    sample_information_table = \
        analyse_mass_spectrometry_data.get_sample_information_table(
        args.sample_information_file_path)

    # Remove not used samples from information table
    label_data_table = analyse_mass_spectrometry_data.get_label_data_table(
        args.label_data)

    # Write predictions to sample information table
    sample_information_table = update_sample_information_table(
        label_data_table, sample_information_table)
    # Export heatmaps
    export_heatmaps(args.output_file, sample_information_table)
    return 0


if __name__ == "__main__":
    sys.exit(main())
```

Python file mass_spectrum_analyzer.py:

```
#!/usr/bin/env python3
"""
Script containing a mass spectra analyzer class responsible for
analysing mass spectra in mzXML format
"""

import os
```

```
import re
import sys

from sample import Sample

__author__ = "C.A. (Robert) Warmerdam"
__credits__ = ["Robert Warmerdam"]
__version__ = "0.0.2"
__status__ = "Development"


class MassSpectraAnalyzer:
    def __init__(self, data, sample_information_table,
                 retention_time_range, mz_range):
        self.samples = self.get_spectra(data, sample_information_table,
                                        retention_time_range, mz_range)
        self.sample_information_table = sample_information_table.copy(
            deep=True)

    @staticmethod
    def get_spectra(data, sample_information_table,
                    retention_time_range, mz_range):
        """
        Creates samples by matching the data file names with the
        samples information table data
        The location from the samples information table should be
        present in a file name
        :param retention_time_range:
        :param mz_range:
        :param data: list of mzXML files
        :param sample_information_table: pandas data frame containing
        at least the columns 'location' and 'mol_weight'
        :return: list of samples
        """
        samples = list()
        for file in data:

            path = file.name
            filename = os.path.basename(path)

            # Select the row from the samples information table that
            # is present in the filename
            row = sample_information_table[
                [re.search(loc + r'\D', filename) is not None for loc in
                 sample_information_table['location']]]

            # Get molecular weight
            mol_weight = float(row['mol_weight'])

            # Create spectrum instance
            identifier = row['location'].values[0]

            # Add the spectrum if the molecular weight is within the
            # required range
            if mz_range[0] <= mol_weight <= mz_range[1]:
                print(
                "\rImporting samples, file: {}, {}".format(identifier,
```

```
                                                filename),
                    end='')
                    samples.append(Sample(path, identifier, mol_weight,
                                          retention_time_range, mz_range))
                else:
                    print(
                    "{}The mass of the product in well '{}' does not fit "
                    "in the given mz range '{}-{}'".format(
                        os.linesep,
                        identifier,
                        *mz_range
                    ), file=sys.stderr, flush=True)
            print(os.linesep)
            return samples

        def process_spectra(self):
            """
            Process mass spectra samples
            """
            print("Processing spectra...")
            for sample in self.samples:
                sample.get_peak_spectrum()
                sample.process_mass_spectrum()
                sample.get_prediction()

        def get_results_table(self):
            """
            Process results to include the prediction and filter non
            predicted entries
            :return: sample information numpy table with prediction column
            """
            # Remove not used samples from information table
            ids = [spectrum.identifier for spectrum in self.samples]
            sample_information_table = self.sample_information_table[
                self.sample_information_table['location'].isin(ids)].copy()

            # Write predictions to sample information table
            for index, row in sample_information_table.iterrows():
                sample = list(
                    filter(lambda x: x.identifier == row['location'],
                           self.samples))[0]
                sample_information_table.at[
                    index, 'prediction'] = sample.prediction
            return sample_information_table


    def main(argv=None):
        if argv is None:
            argv = sys.argv
        return 0


    if __name__ == "__main__":
        sys.exit(main())
```

Python file sample.py:

```
#!/usr/bin/env python3
"""
Script containing a Sample class that can be used for scoring the sample
"""

import base64
import struct
import xml
import zlib
import sys
from decimal import Context

from collections import Counter

import numpy as np
from matplotlib import pyplot

__author__ = "C.A. (Robert) Warmerdam"
__credits__ = ["Robert Warmerdam"]
__status__ = "Development"

MZ_XML_SCAN_ELEMENT = './/{' \
                      'http://sashimi.sourceforge.net/schema_revision' \
                      '/mzXML_3.2}scan'
MZ_ERROR = 0.5
PLOT = False


class Sample:
    non_frequent_adduct_masses = [
        ("M+3H", 0.33, 1.007276),
        ("M+2H+Na", 0.33, 8.334590),
        ("M+H+2Na", 0.33, 15.766190),
        ("M+3Na", 0.33, 22.989218),
        ("M+2H", 0.5, 1.007276),
        ("M+H+NH4", 0.5, 9.520550),
        ("M+H+Na", 0.5, 11.998247),
        ("M+H+K", 0.5, 19.985217),
        ("M+ACN+2", 0.5, 21.520550),
        ("M+2Na", 0.5, 22.989218),
        ("M+2ACN+2", 0.5, 42.033823),
        ("M+3ACN+2", 0.5, 62.547097)
    ]
    frequent_adduct_masses = [
        ("M+H", 1, 1.007276),
        ("M+NH4", 1, 18.033823),
        ("M+Na", 1, 22.989218),
        ("M+CH3OH+H", 1, 33.033489),
        ("M+K", 1, 38.963158),
        ("M+ACN+H", 1, 42.033823),
        ("M+2Na-H", 1, 44.971160),
        ("M+IsoProp+H", 1, 61.065340),
        ("M+ACN+Na", 1, 64.015765),
        ("M+2K+H", 1, 76.919040),
        ("M+DMSO+H", 1, 79.021220),
```

```
    ("M+2ACN+H", 1, 83.060370),
    ("M+IsoProp+Na+H", 1, 84.055110),
    ("2M+H", 2, 1.007276),
    ("2M+NH4", 2, 18.033823),
    ("2M+Na", 2, 22.989218),
    ("2M+3H2O+2H", 2, 28.023120),
    ("2M+K", 2, 38.963158),
    ("2M+ACN+H", 2, 42.033823),
    ("2M+ACN+Na", 2, 64.015765)]
ctx = Context(prec=10)

def __init__(self, path, identifier, mol_weight,
             retention_time_range, mz_range):
    self.mass_array = None
    self.intensity_array = None
    self.data = Counter()
    self.spectrum_tree = None
    try:
        # Try to open the file and read it
        self.spectrum_tree = xml.etree.ElementTree.parse(path)
    except xml.etree.ElementTree.ParseError as e:
        raise Exception("{} in '{}'".format(str(e), path))
    self.retention_time_range = retention_time_range
    self.mol_weight = mol_weight
    self.identifier = identifier
    self.path = path
    self.mz_range = list(mz_range)
    # self.mz_range[0] = self.mol_weight
    self._prediction = 0

@property
def prediction(self):
    return self._prediction

@staticmethod
def _decode_peaks(content, d_array_length):
    """
    Decode peaks
    :param content: the encoded array
    :param d_array_length: amount of peaks
    :return: list of decoded values with alternating m/z value
    and intensity
    """
    # Decode base64 data
    decoded_data = base64.b64decode(content)

    # Decompress data
    uncompressed_data = zlib.decompress(decoded_data)

    # Unpack data
    data = struct.unpack('>{}d'.format(d_array_length * 2),
                         uncompressed_data)
    return data

def _get_frequent_adduct_masses(self):
    """
    Calculate masses with adduct entries.
```

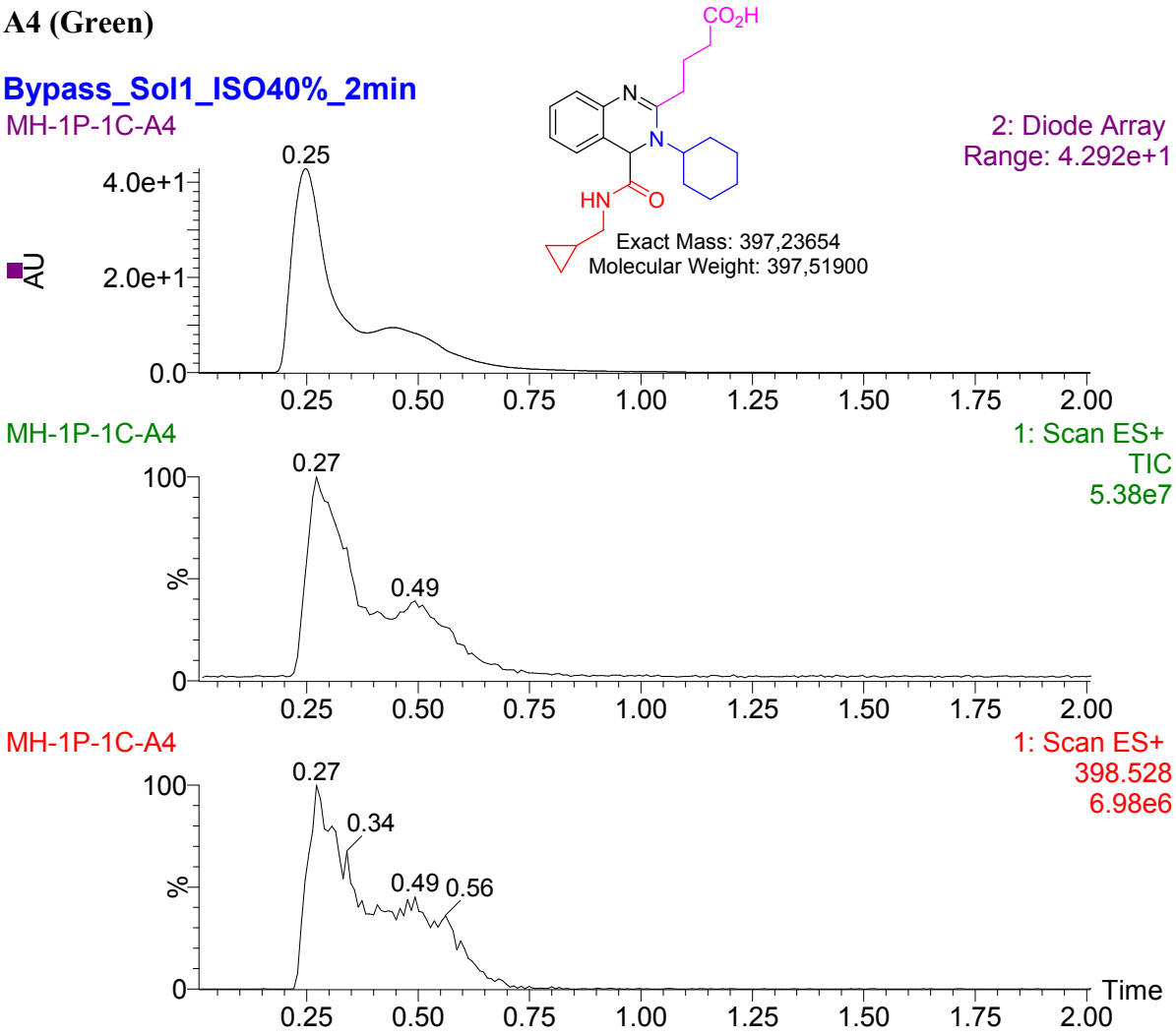
```
    :return: masses sorted by distance between mol weight (
    ascending)
    """
    masses = [(self.mol_weight * multiplier + addition, iso) for
              iso, multiplier, addition in
              self.frequent_adduct_masses]
    ranked_masses = list()
    for i, mass in enumerate(sorted(masses, key=lambda m: abs(
            self.mol_weight - m[0]))):
        # Round mass as instructed
        ranked_masses.append((round(mass[0]), mass[1]))
    return ranked_masses

def get_peak_spectrum(self):
    """
    Get the peak spectrum
    """
    mass_list = []
    intensity_list = []
    # Collect scans in the time range and with msLevel not 0
    for scan in self.spectrum_tree.findall(MZ_XML_SCAN_ELEMENT):

        # Make sure PT...  and ...S are stripped from retention
        # time attribute
        # Divide by 60 to get seconds
        retention_time = float(
            scan.attrib['retentionTime'][2:-1]) / 60
        # Discard the scan when msLevel is 0 or when the scan is
        # not within time range
        if scan.attrib['msLevel'] in ["0", ""] or \
                not (self.retention_time_range[0] < retention_time <
                     self.retention_time_range[1]):
            continue

        # Get the peaks
        peaks = scan.find(
            '{http://sashimi.sourceforge.net/schema_revision'
            '/mzXML_3.2}peaks')
        d_array_length = int(scan.attrib['peaksCount'])
        # Decode the peaks
        peaks = self._decode_peaks(peaks.text, d_array_length)

        # Extend lists
        mass_list.extend(peaks[::2])
        intensity_list.extend(peaks[1::2])
    # Filter the NAN and infinity values from the peak spectrum
    mass_array = np.asarray(mass_list)
    intensity_array = np.asarray(intensity_list)
    indices = np.where(
        np.logical_or(np.isnan(mass_array), np.isinf(mass_array),
                      np.isnan(intensity_array)))
    for index in indices:
        mass_array = np.delete(mass_array, index)
        intensity_array = np.delete(intensity_array, index)
    self.mass_array = mass_array
    self.intensity_array = intensity_array
```

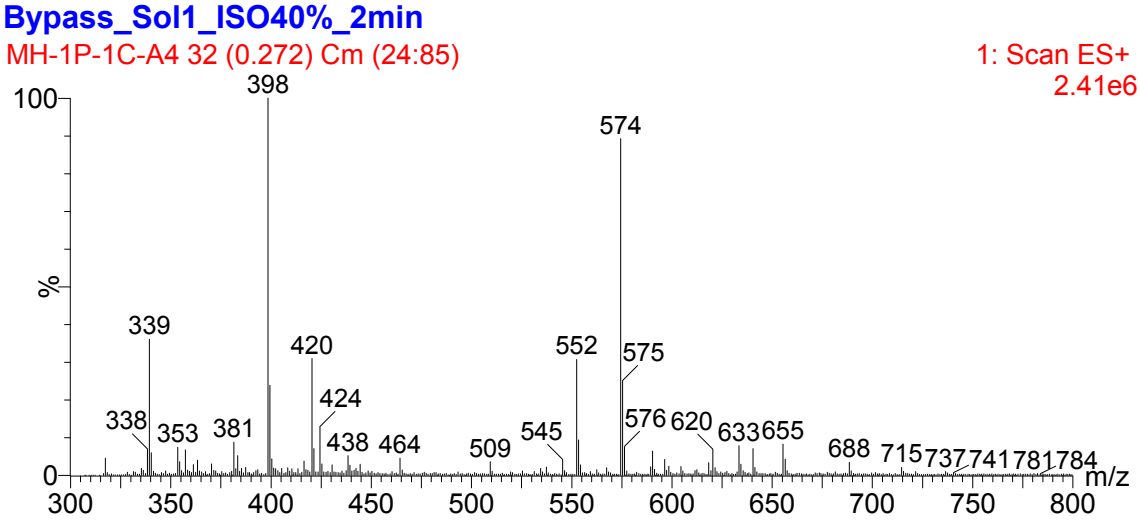
```
def process_mass_spectrum(self):
    """
    Process mass spectrum
    """
    mz_range = range(int(self.mz_range[0]), int(self.mz_range[1]))

    # Get inter quantile range
    q1 = np.nanquantile(self.intensity_array, 0.25)
    q3 = np.nanquantile(self.intensity_array, 0.75)
    iqr = q3 - q1

    for i, mz in enumerate(self.mass_array):
        # Filter out every intensity value below 0 or above
        # quantile q3 + 5000 * inter quantile range
        if self.intensity_array[i] < 0 or self.intensity_array[
            i] > (q3 + 5000 * iqr):
            # Next
            continue
        for mz_value in mz_range:
            # Collect peaks with max 0.5 difference between
            # integer MZ value
            if (mz_value - MZ_ERROR) <= mz <= (mz_value + MZ_ERROR):
                self.data[mz_value] += self.intensity_array[i]

def get_prediction(self):
    """
    Get the prediction of this sample
    """
    mz_list, int_list = tuple(
        zip(*list(sorted(self.data.items(), key=lambda x: x[0]))))

    # Plot if PLOT set to True
    if PLOT:
        self.plot_spectrum(int_list, mz_list)

    # Get frequent adduct masses to look for
    mz_s = self._get_frequent_adduct_masses()

    # Print important values
    print("Predicting: {}".format(self.identifier))
    print('exact mass:', self.mol_weight)
    # Get maximum intensity (base peak intensity)
    max_intensity = max(list(self.data.values()))
    print("highest peak:", mz_list[int_list.index(max_intensity)],
          max_intensity)

    # Initialize mz of final score
    final_mz = None
    for mz, ion_name in mz_s:
        # Get relative intensity
        intensity = self.data[mz]
        rel_intensity = intensity / max_intensity
        score = rel_intensity
        # Check if this scores better
        if score > self._prediction:
            self._prediction = score
            final_mz = mz, ion_name
```

```
        if final_mz is not None:
            print("{} Prediction: {} at M/Z {} ({})".format(
                self.identifier, self.prediction, *final_mz))

    def plot_spectrum(self, int_list, mz_list):
        """
        Method that plots the spectrum in a histogram, with m/z
        identifier above each bar
        :param int_list: The list of intensities
        :param mz_list: The list of mass-charge ratios
        """
        pyplot.title(self.identifier + ": " + str(self.mol_weight))
        rects = pyplot.bar(np.asarray(mz_list), int_list, 1)
        pyplot.xlabel("m/z")
        pyplot.ylabel("Intensity")
        for i, rect in enumerate(rects):
            # Get the height of the bar
            height = rect.get_height()
            # Plot the m/z above each bar
            pyplot.text(rect.get_x() + rect.get_width() / 2.,
                        1.05 * height,
                        mz_list[i],
                        ha='center', va='bottom', fontsize=6)
        pyplot.show()


def main(argv=None):
    if argv is None:
        argv = sys.argv
    return 0


if __name__ == "__main__":
    sys.exit(main())
```

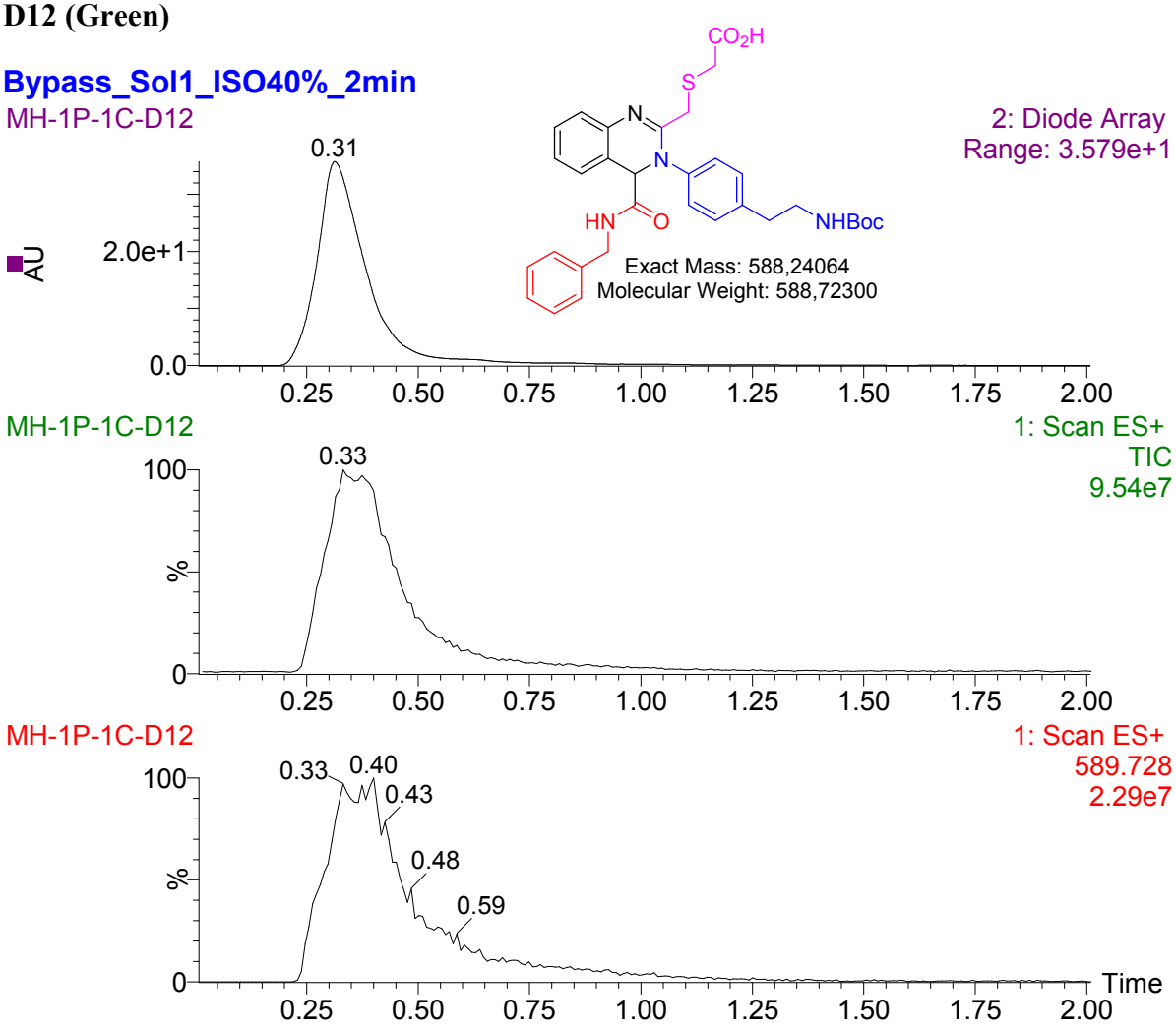
Examples of SFC-MS analytics directly out of the 384-well plate

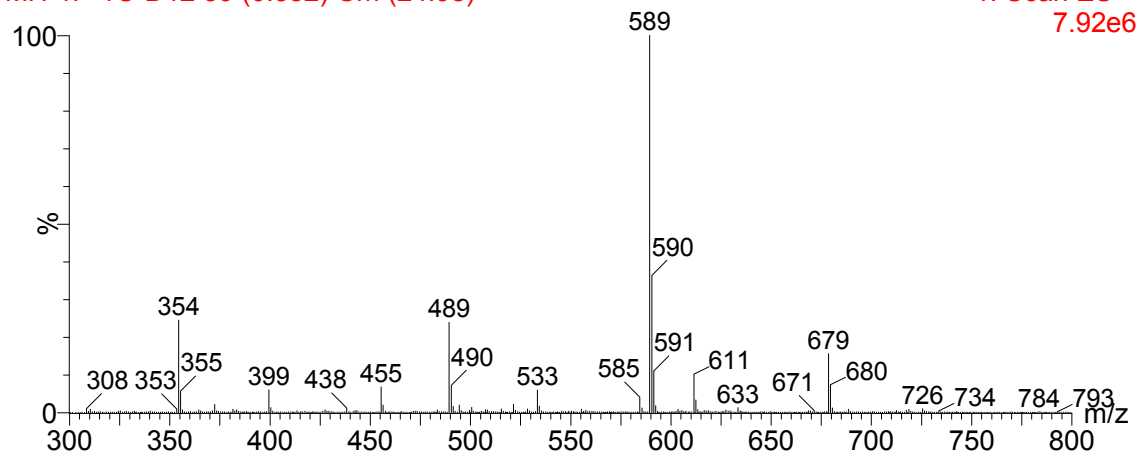
A4 (Green)



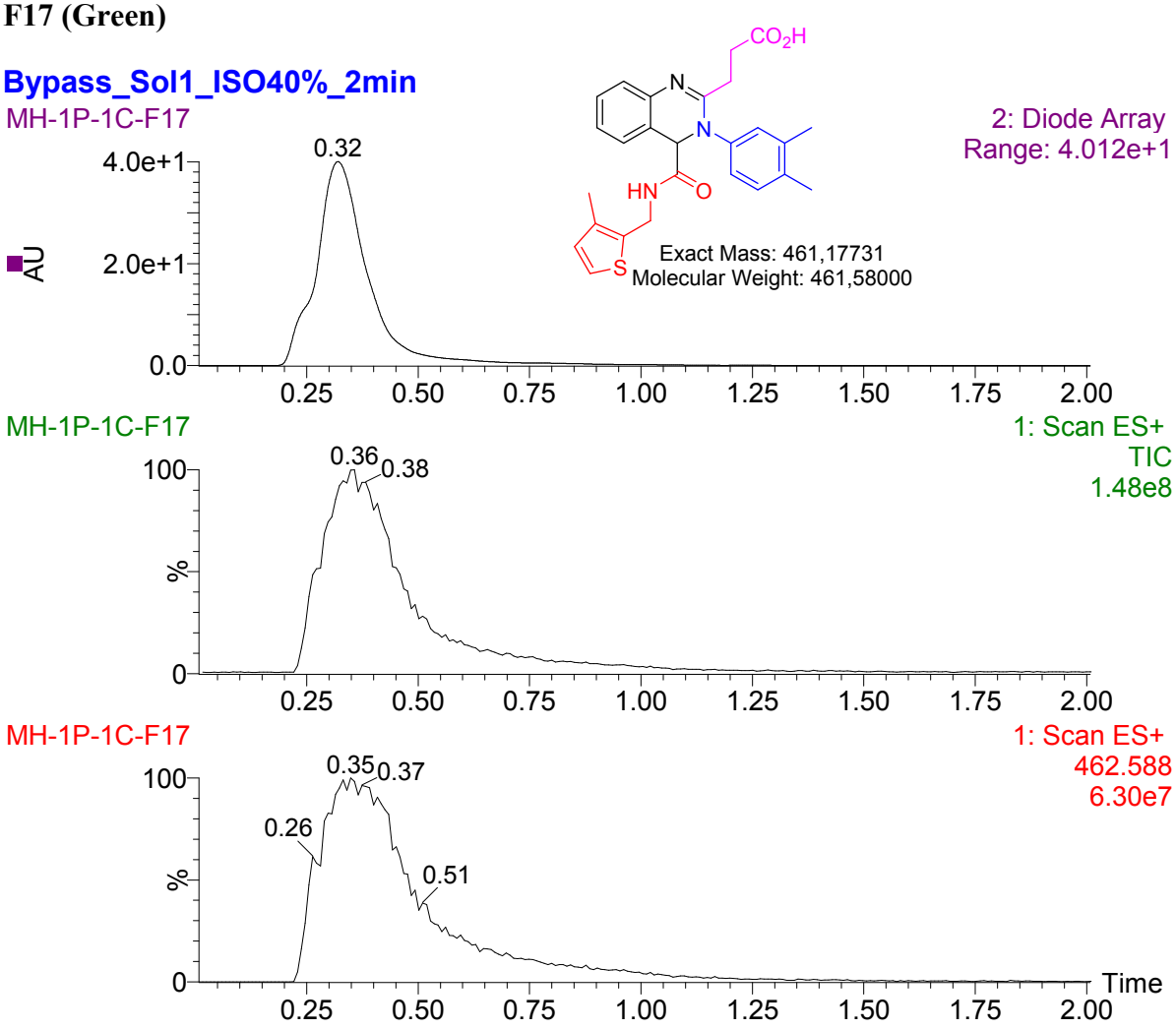


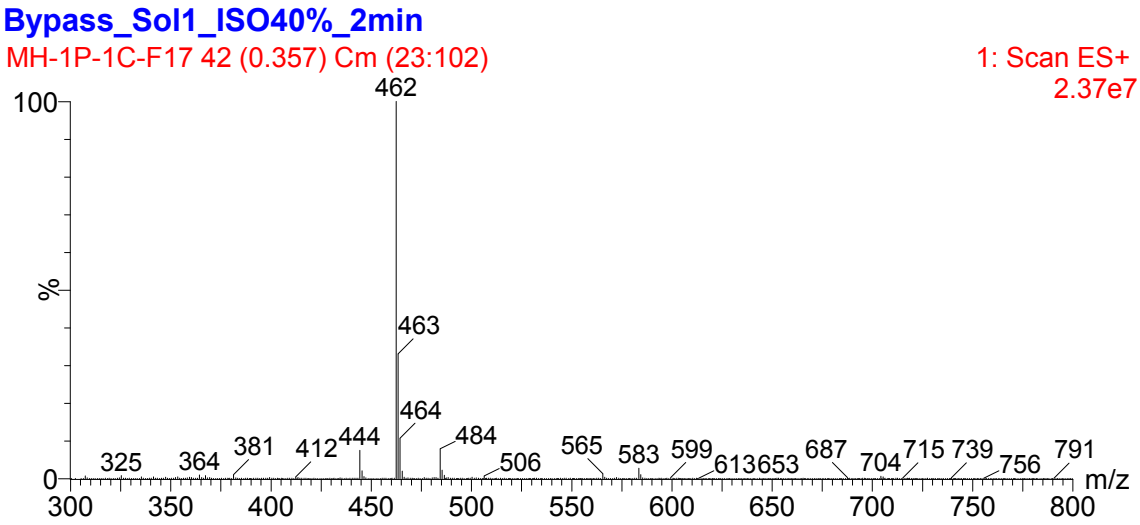
D12 (Green)


Bypass_Sol1_ISO40%_2min
MH-1P-1C-D12
2: Diode Array
Range: 3.579e+1
0.31
AU
2.0e+1
0.0
CO2H
S
N
N
HN
O
NHBoc
Exact Mass: 588,24064
Molecular Weight: 588,72300
0.25 0.50 0.75 1.00 1.25 1.50 1.75 2.00
MH-1P-1C-D12
1: Scan ES+
TIC
9.54e7
0.33
100
%
0
MH-1P-1C-D12
1: Scan ES+
589.728
2.29e7
0.33 0.40 0.43 0.48 0.59
Time

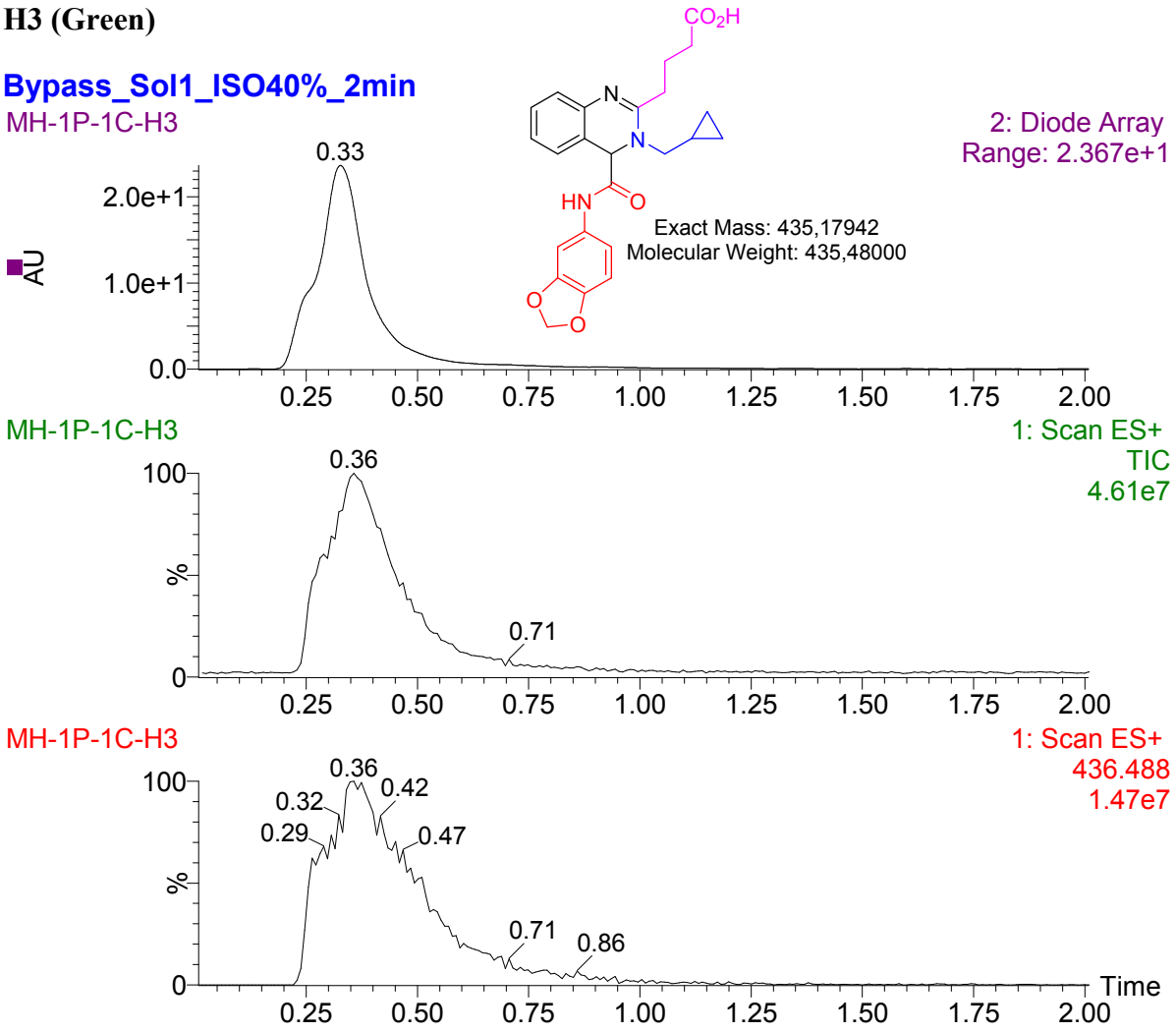

Bypass_Sol1_ISO40%_2min
MH-1P-1C-D12 39 (0.332) Cm (24:98)
1: Scan ES+
7.92e6
100
%
0
589
590
591
354
355
489
490
679
680
611
633
671
308 353
399 438 455
533
585
726 734 784 793
m/z
300 350 400 450 500 550 600 650 700 750 800

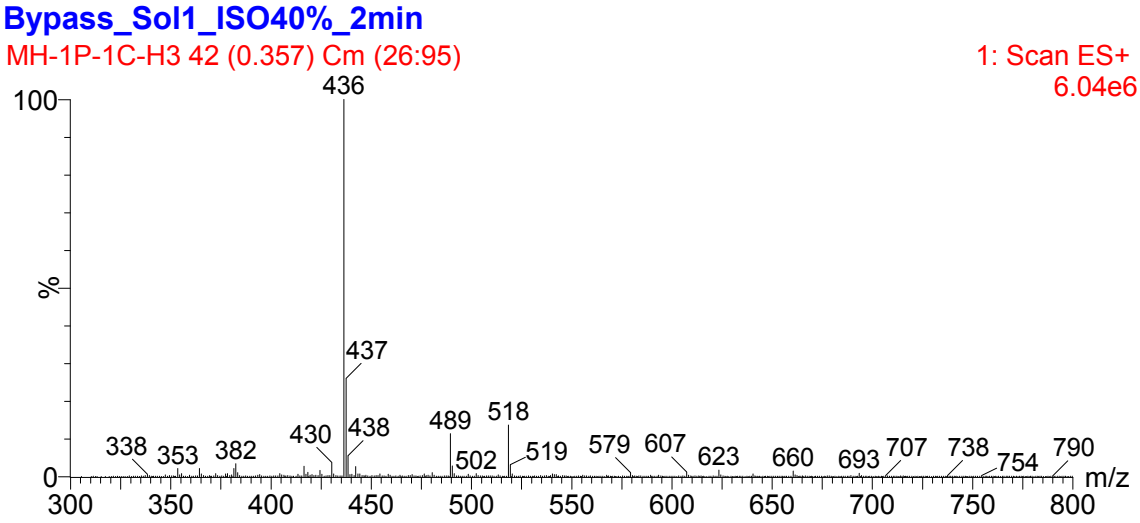
F17 (Green)


Bypass_Sol1_ISO40%_2min
MH-1P-1C-F17
2: Diode Array
Range: 4.012e+1
0.32
4.0e+1
2.0e+1
0.0
AU
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
CO2H
N
N
HN
O
S
Exact Mass: 461,17731
Molecular Weight: 461,58000
MH-1P-1C-F17
1: Scan ES+
TIC
1.48e8
0.36
0.38
100
%
0
MH-1P-1C-F17
1: Scan ES+
462.588
6.30e7
0.35
0.37
0.26
0.51
Time

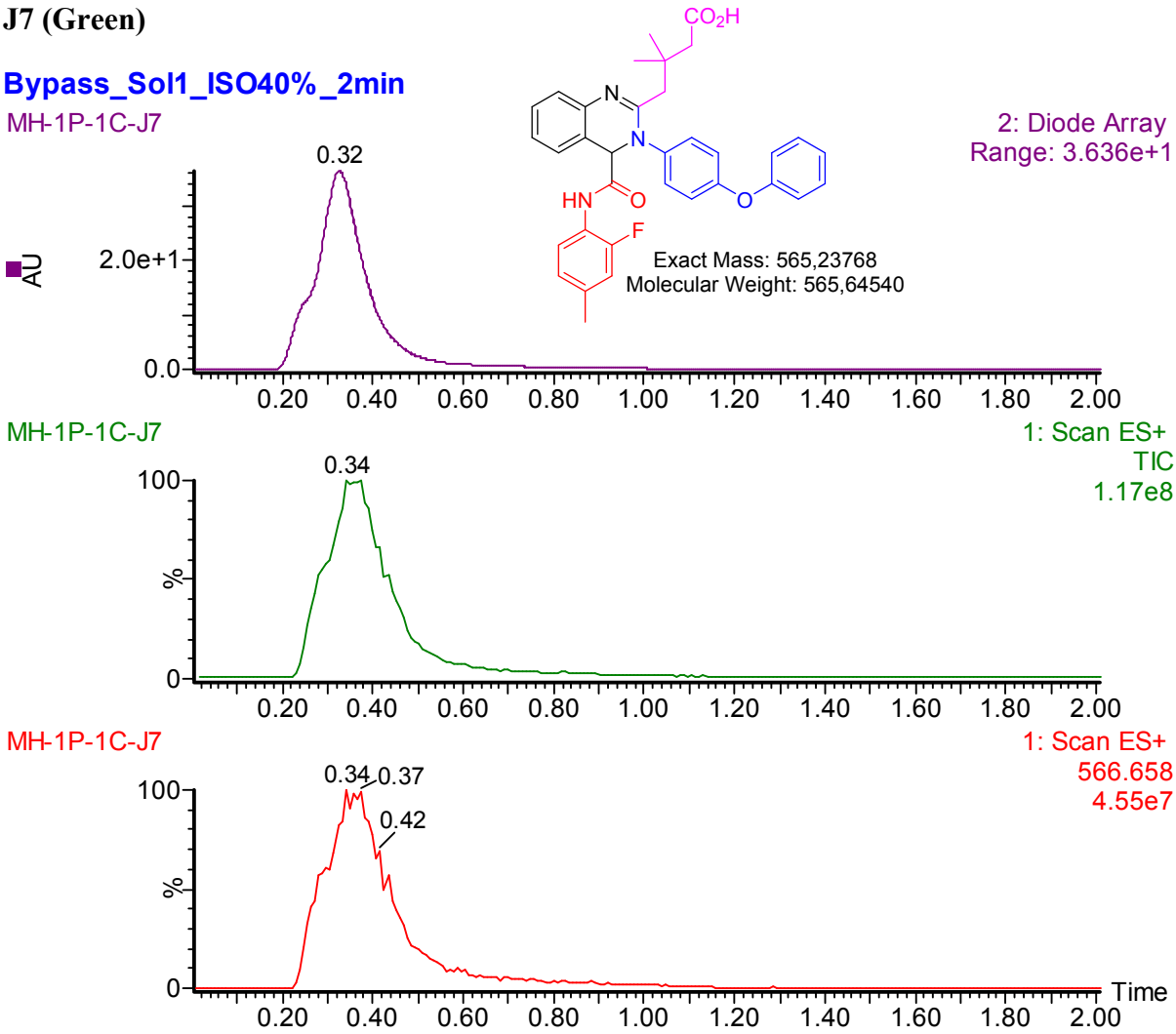

Bypass_Sol1_ISO40%_2min
MH-1P-1C-F17 42 (0.357) Cm (23:102)
1: Scan ES+
2.37e7
100
%
0
462
463
464
325
364
381
412
444
484
506
565
583
599
613653
687
704
715
739
756
791
300
350
400
450
500
550
600
650
700
750
800
m/z

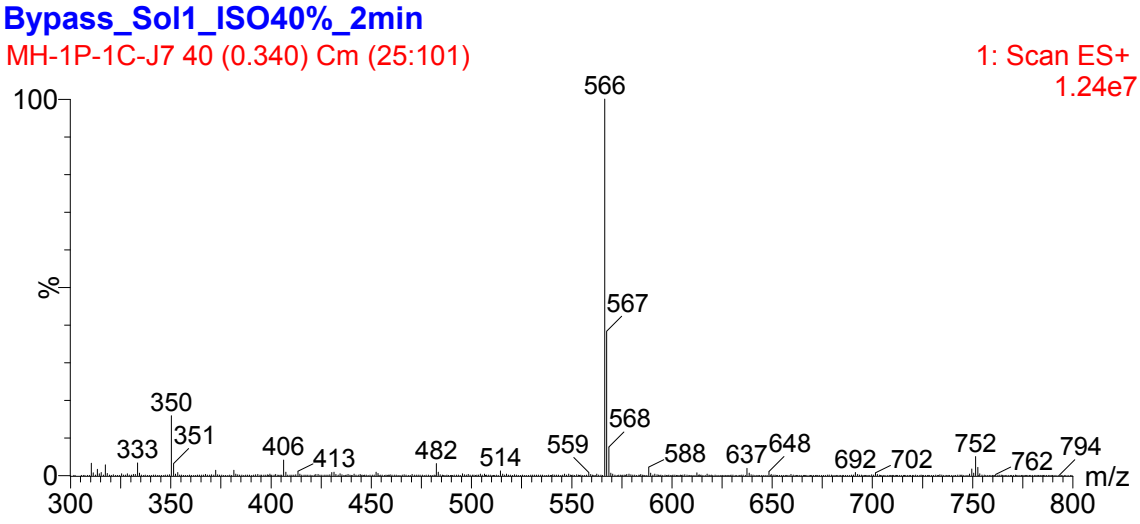
H3 (Green)


Bypass_Sol1_ISO40%_2min
MH-1P-1C-H3
2: Diode Array
Range: 2.367e+1
0.33
2.0e+1
1.0e+1
0.0
AU
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
CO2H
N
N
HN
O
O
O
Exact Mass: 435,17942
Molecular Weight: 435,48000
MH-1P-1C-H3
1: Scan ES+
TIC
4.61e7
0.36
100
%
0
0.71
MH-1P-1C-H3
1: Scan ES+
436.488
1.47e7
0.36
0.42
0.32
0.29
0.47
0.71
0.86
Time

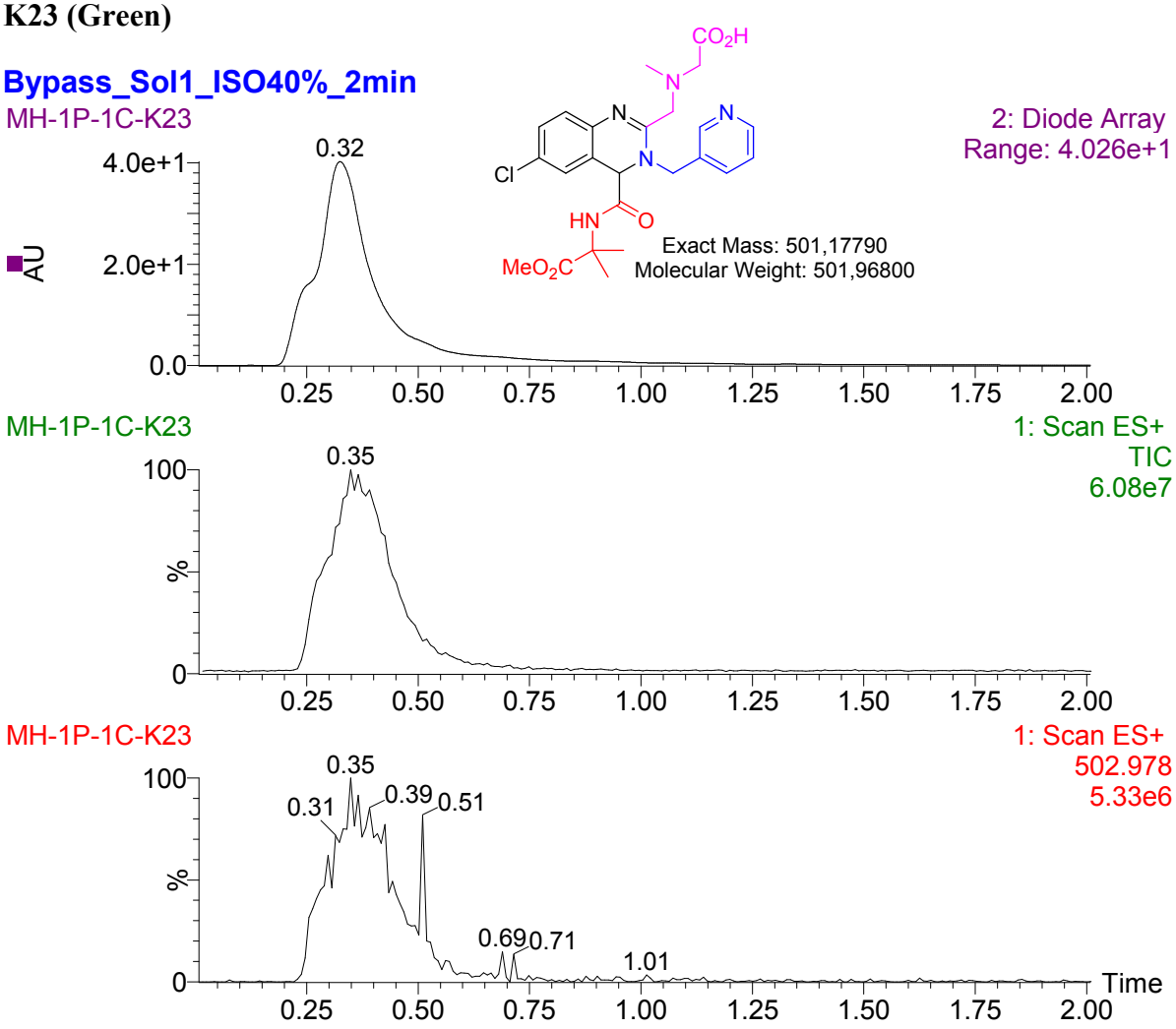

Bypass_Sol1_ISO40%_2min
MH-1P-1C-H3 42 (0.357) Cm (26:95)
1: Scan ES+
6.04e6
436
100
%
0
437
438
430
338
353
382
489
518
502
519
579
607
623
660
693
707
738
754
790
m/z
300
350
400
450
500
550
600
650
700
750
800

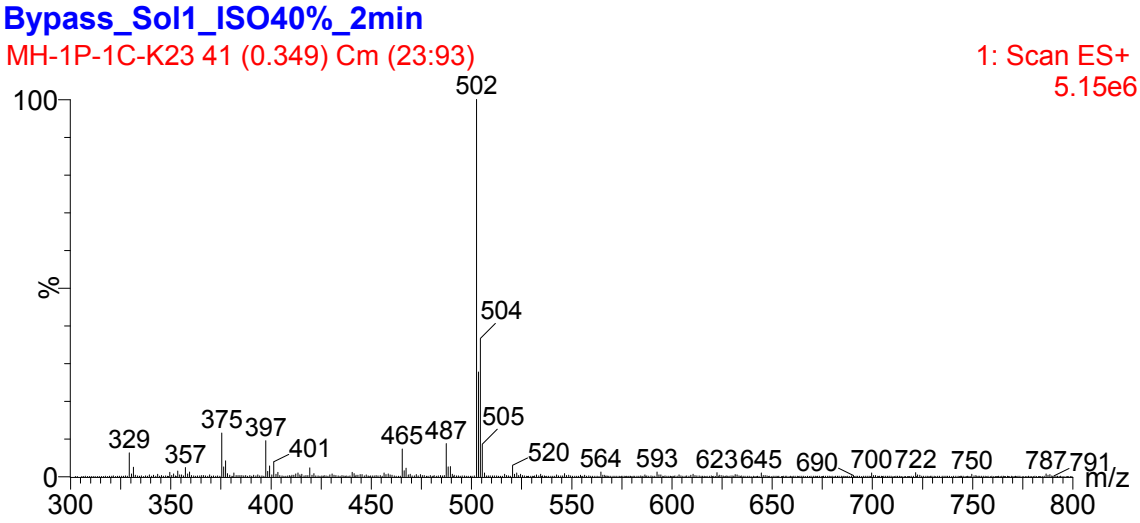
J7 (Green)


Bypass_Sol1_ISO40%_2min
MH-1P-1C-J7
2: Diode Array
Range: 3.636e+1
0.32
AU
2.0e+1
0.0
Exact Mass: 565,23768
Molecular Weight: 565,64540
0.20 0.40 0.60 0.80 1.00 1.20 1.40 1.60 1.80 2.00
MH-1P-1C-J7
1: Scan ES+
TIC
1.17e8
0.34
100
%
0
MH-1P-1C-J7
1: Scan ES+
566.658
4.55e7
0.34 0.37
0.42
Time

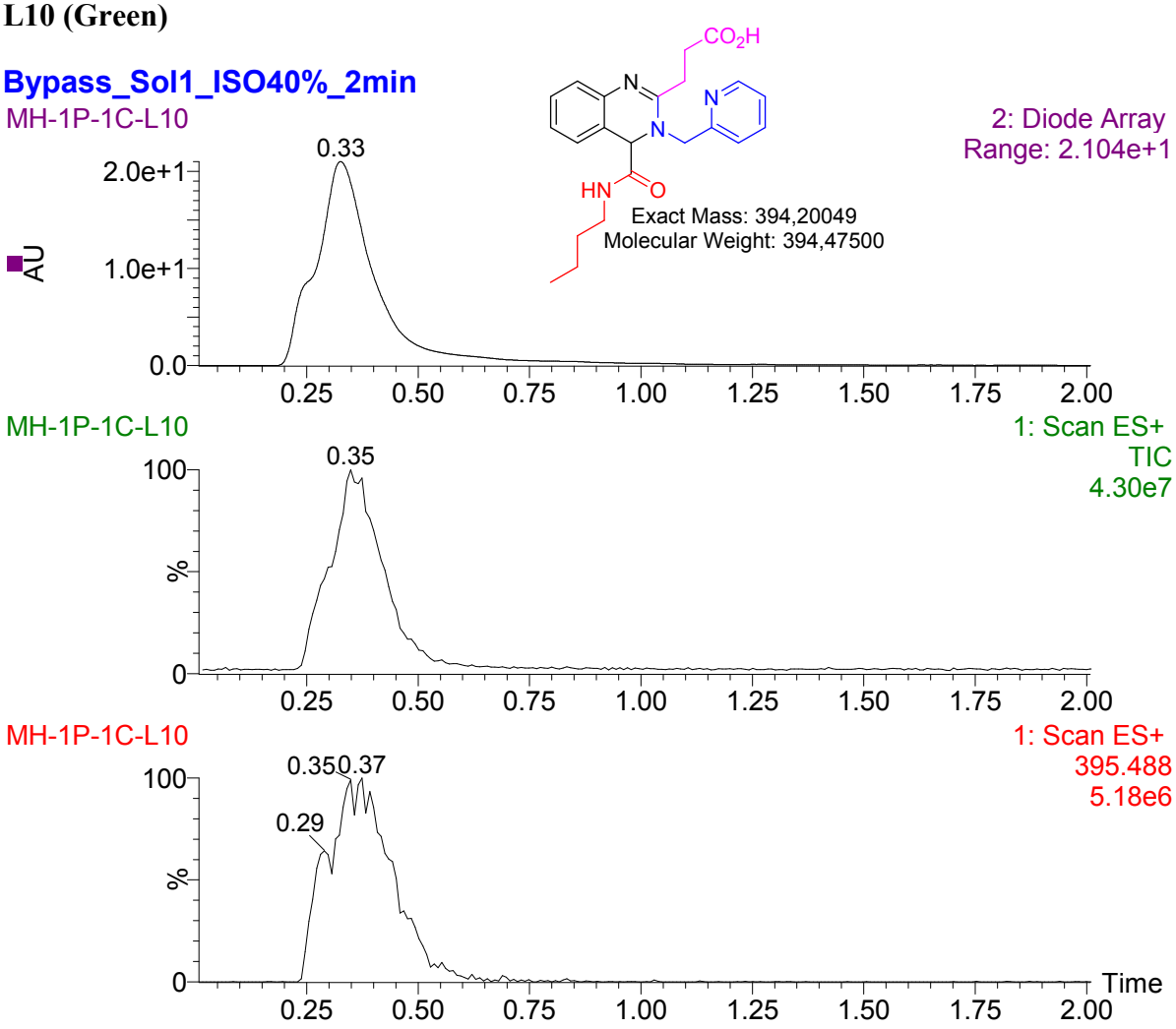

Bypass_Sol1_ISO40%_2min
MH-1P-1C-J7 40 (0.340) Cm (25:101)
1: Scan ES+
1.24e7
100
%
0
566
567
568
350
351
333
406
413
482
514
559
588
637
648
692
702
752
762
794
m/z
300 350 400 450 500 550 600 650 700 750 800

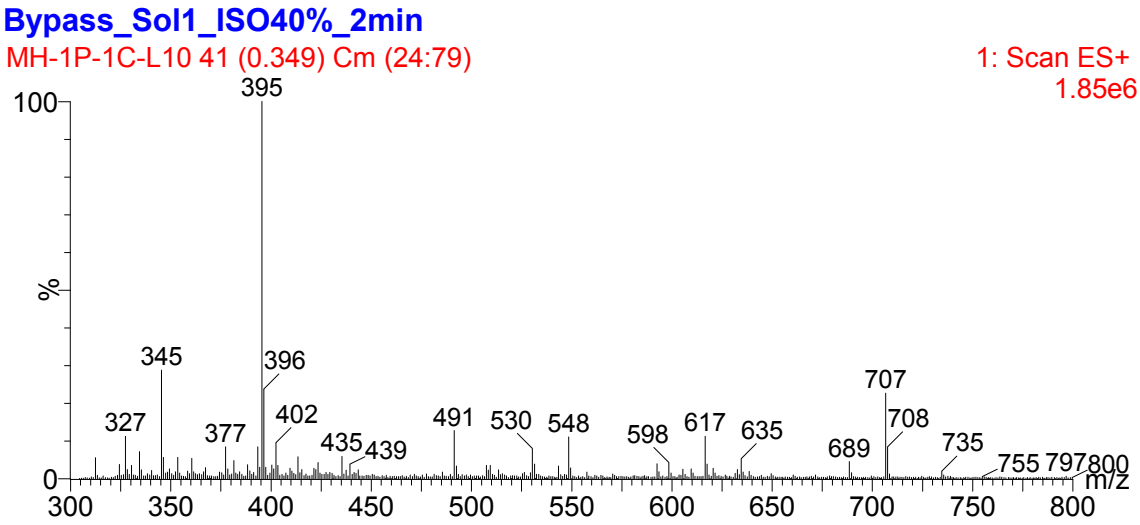
K23 (Green)


Bypass_Sol1_ISO40%_2min
MH-1P-1C-K23
2: Diode Array
Range: 4.026e+1
Exact Mass: 501,17790
Molecular Weight: 501,96800
AU
MH-1P-1C-K23
1: Scan ES+
TIC
6.08e7
MH-1P-1C-K23
1: Scan ES+
502.978
5.33e6
Time

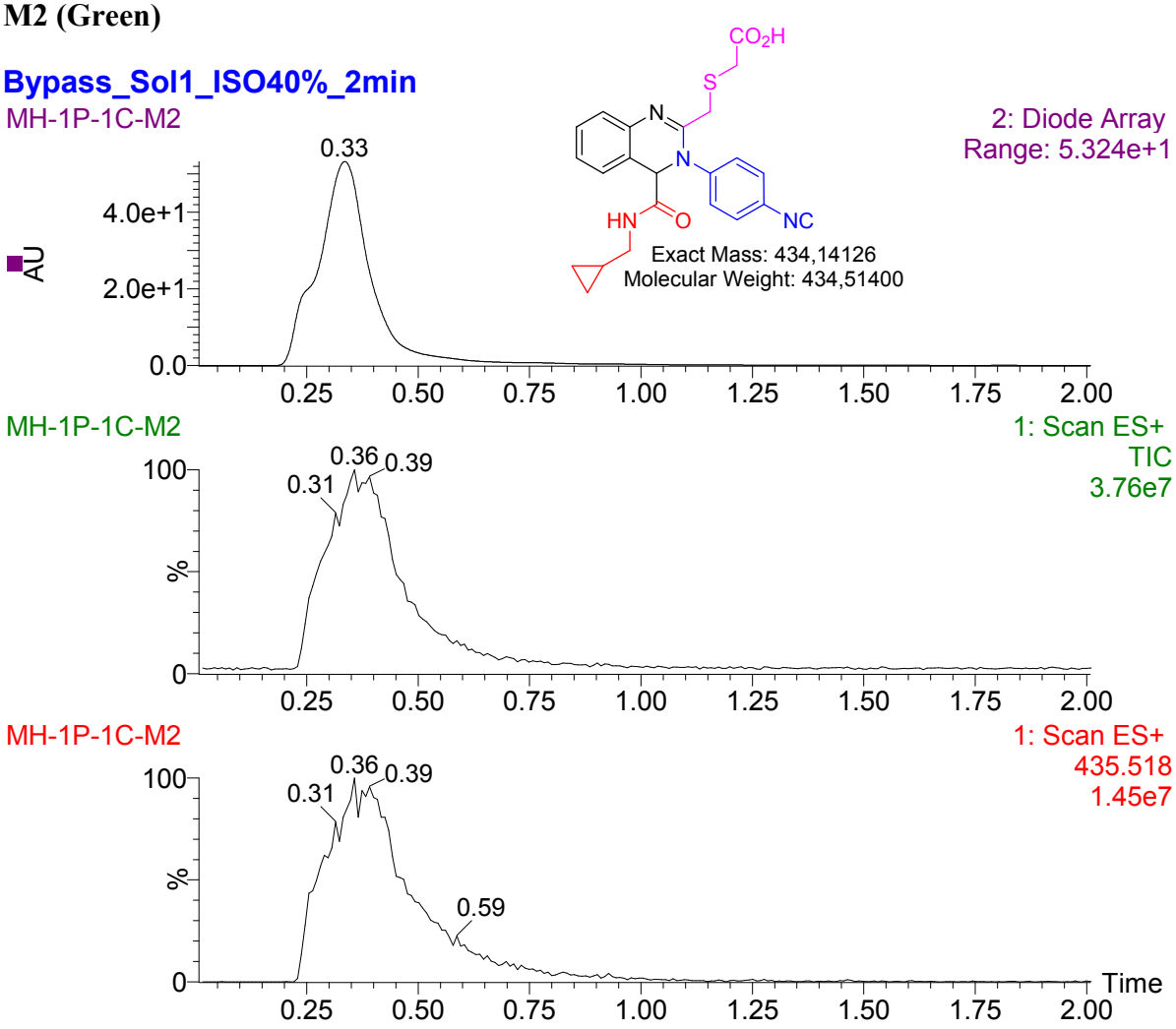

Bypass_Sol1_ISO40%_2min
MH-1P-1C-K23 41 (0.349) Cm (23:93)
1: Scan ES+
5.15e6
m/z

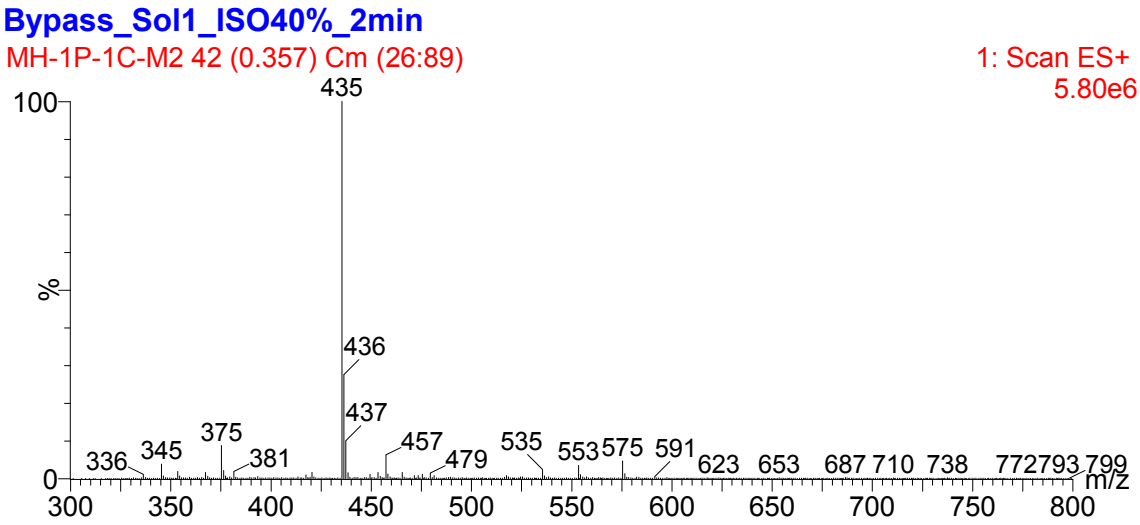
L10 (Green)


Bypass_Sol1_ISO40%_2min
MH-1P-1C-L10
2: Diode Array
Range: 2.104e+1
0.33
AU
Exact Mass: 394,20049
Molecular Weight: 394,47500
MH-1P-1C-L10
1: Scan ES+
TIC
4.30e7
0.35
%
MH-1P-1C-L10
1: Scan ES+
395.488
5.18e6
0.35 0.37
0.29
Time

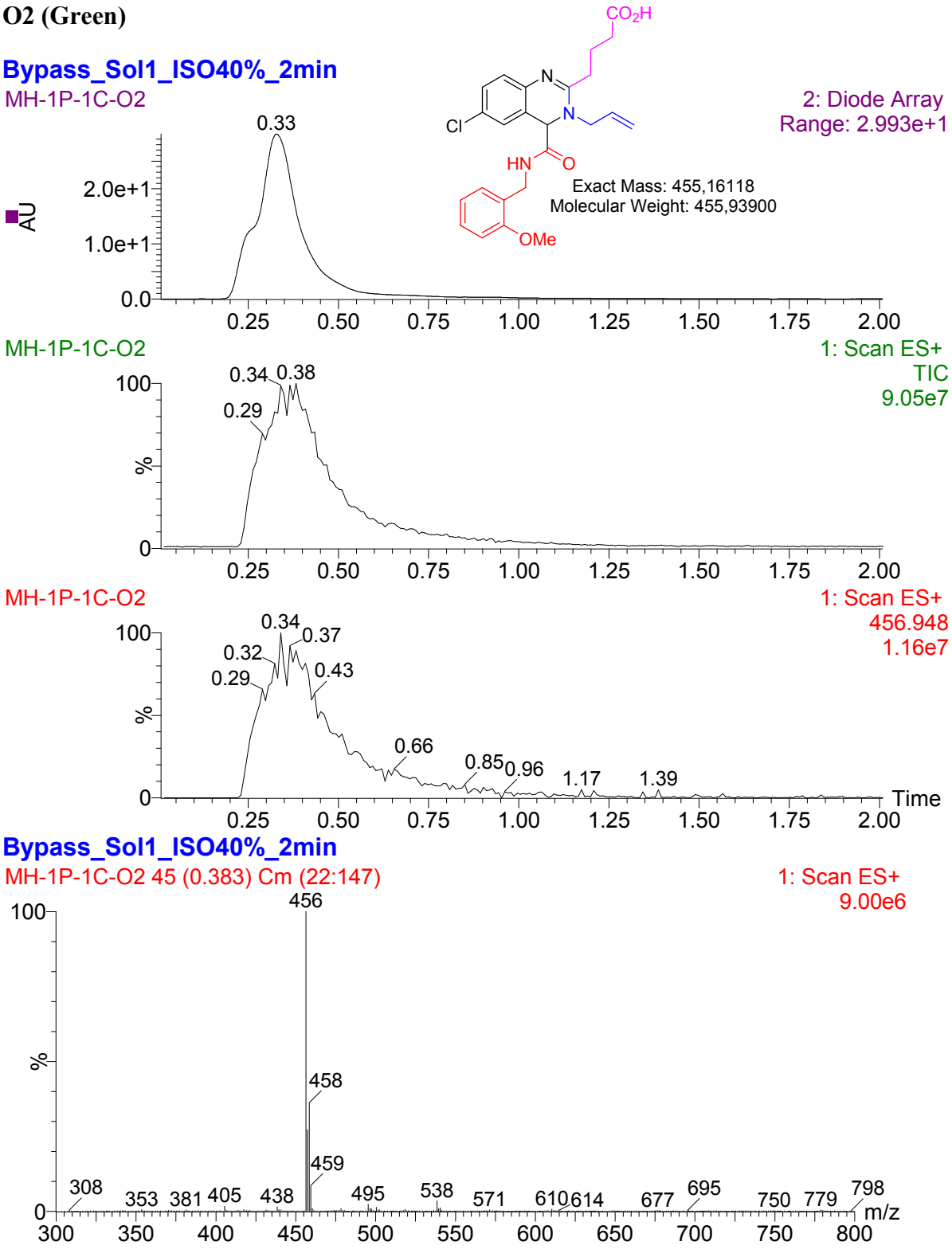

Bypass_Sol1_ISO40%_2min
MH-1P-1C-L10 41 (0.349) Cm (24:79)
1: Scan ES+
1.85e6
395
%
345
396
327
377
402
435
439
491
530
548
598
617
635
689
707
708
735
755
797
800
m/z

M2 (Green)

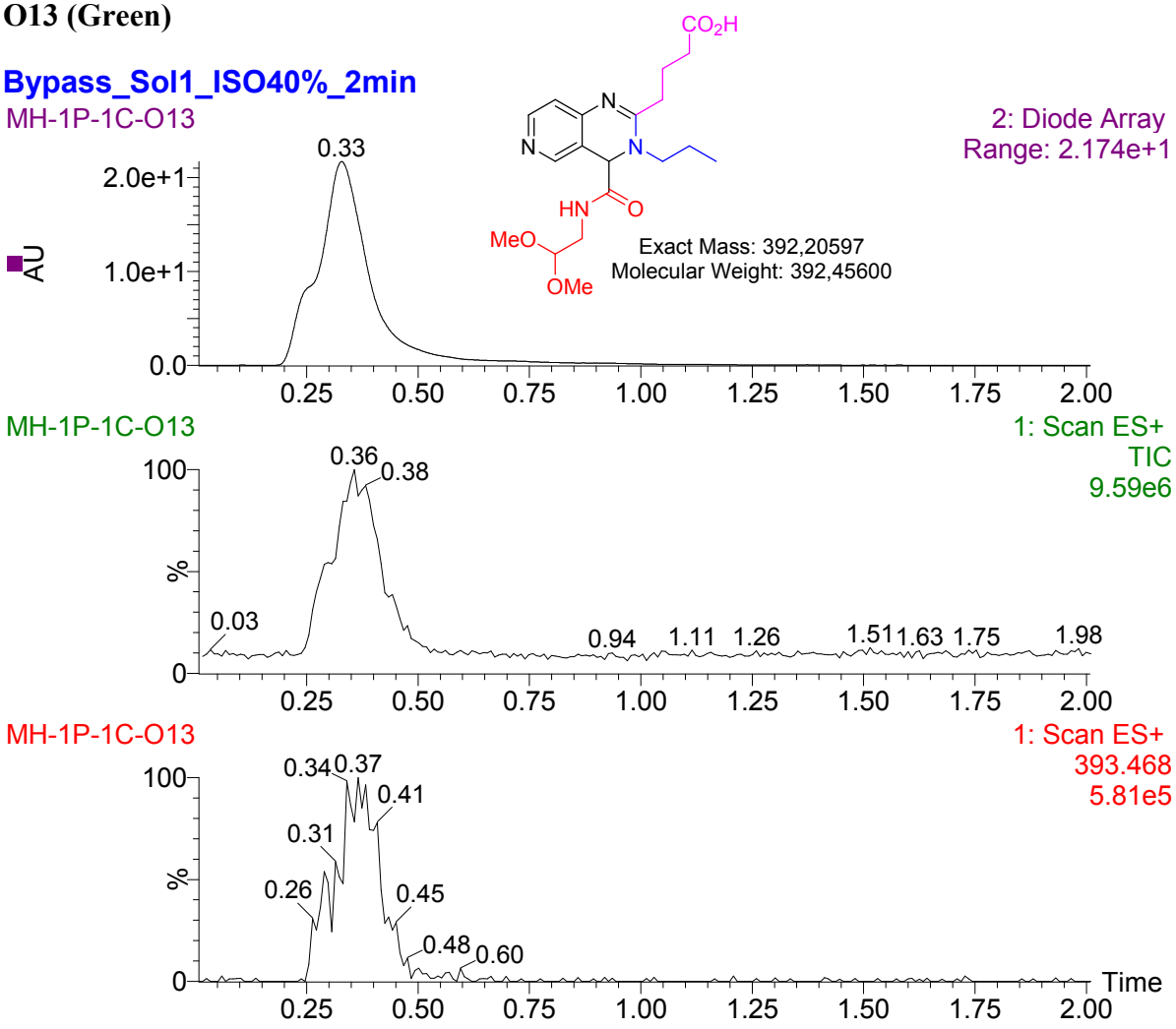

Bypass_Sol1_ISO40%_2min
MH-1P-1C-M2
2: Diode Array
Range: 5.324e+1
0.33
4.0e+1
2.0e+1
0.0
AU
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
CO2H
S
N
N
HN
O
NC
Exact Mass: 434,14126
Molecular Weight: 434,51400
MH-1P-1C-M2
1: Scan ES+
TIC
3.76e7
100
%
0
0.31
0.36
0.39
MH-1P-1C-M2
1: Scan ES+
435.518
1.45e7
0.59
Time

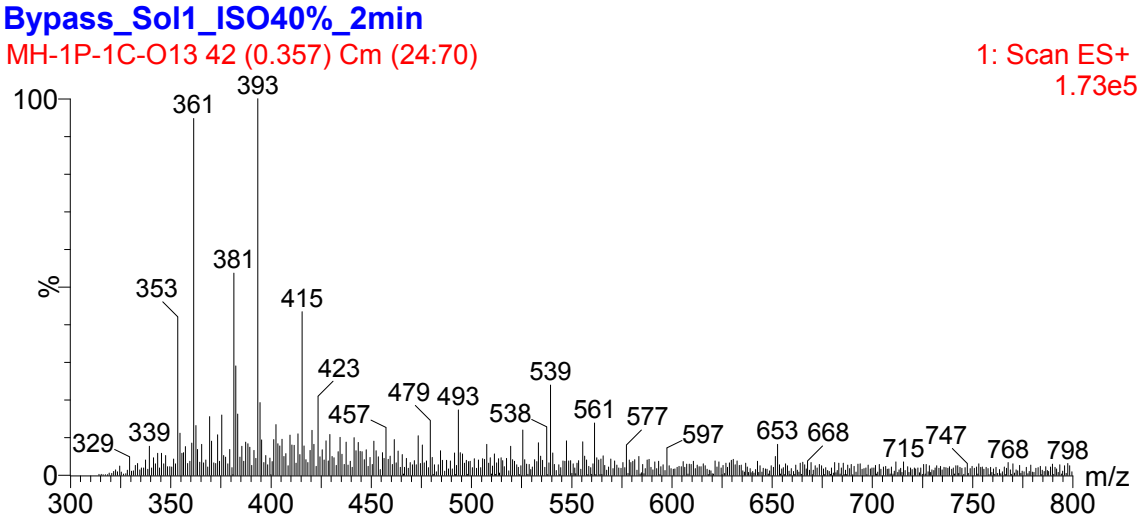

Bypass_Sol1_ISO40%_2min
MH-1P-1C-M2 42 (0.357) Cm (26:89)
1: Scan ES+
5.80e6
100
%
0
435
436
437
336
345
375
381
457
479
535
553
575
591
623
653
687
710
738
772
793
799
m/z
300
350
400
450
500
550
600
650
700
750
800

O2 (Green)

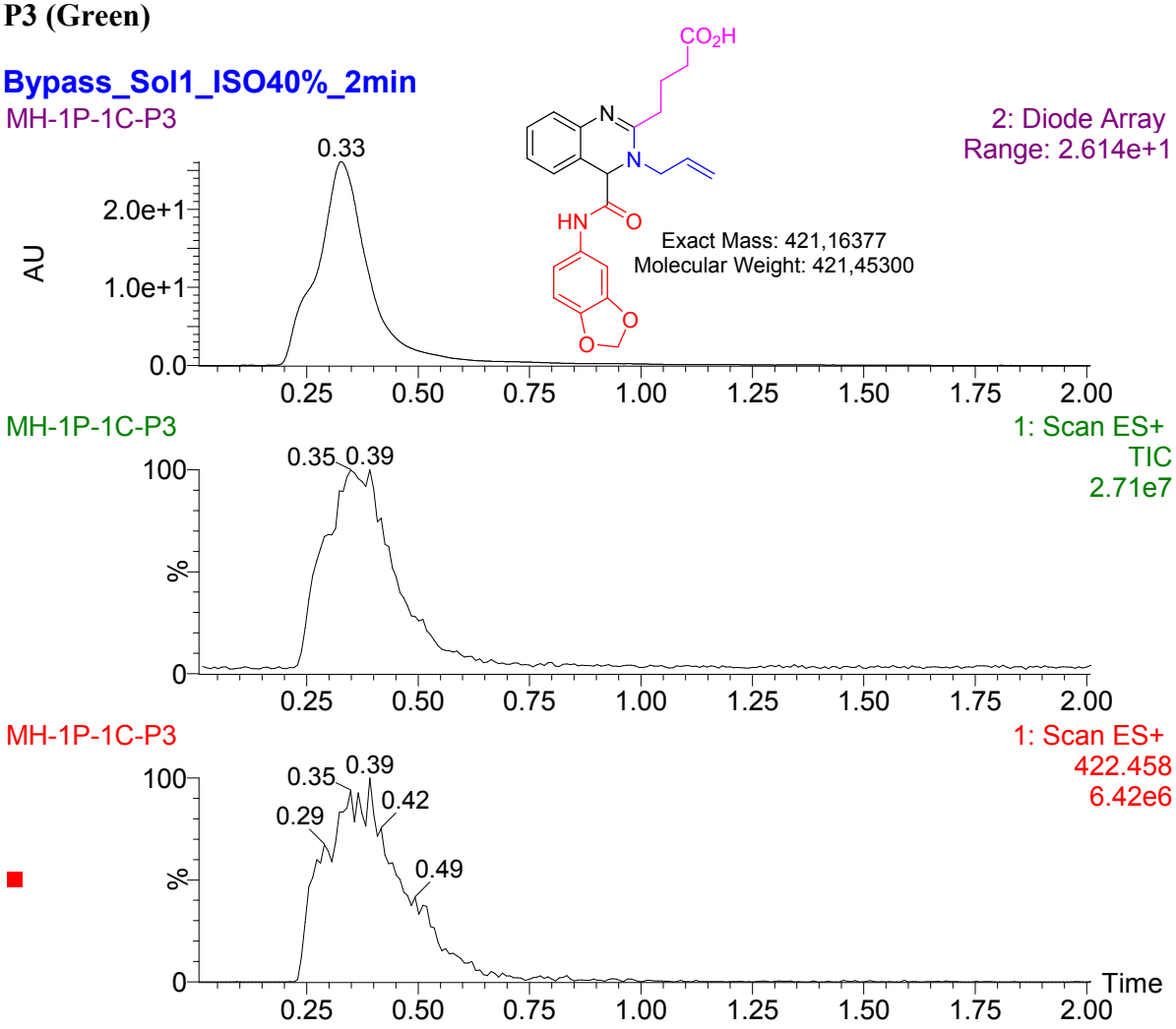

Bypass_Sol1_ISO40%_2min
MH-1P-1C-O2
2: Diode Array
Range: 2.993e+1
0.33
AU
2.0e+1
1.0e+1
0.0
0.25 0.50 0.75 1.00 1.25 1.50 1.75 2.00
CO2H
N
N
Cl
HN
O
OMe
Exact Mass: 455,16118
Molecular Weight: 455,93900
MH-1P-1C-O2
1: Scan ES+
TIC
9.05e7
0.34 0.38
0.29
100
%
0
0.25 0.50 0.75 1.00 1.25 1.50 1.75 2.00
MH-1P-1C-O2
1: Scan ES+
456.948
1.16e7
0.34
0.37
0.32
0.29
0.43
0.66
0.85
0.96
1.17
1.39
100
%
0
Time
0.25 0.50 0.75 1.00 1.25 1.50 1.75 2.00
Bypass_Sol1_ISO40%_2min
MH-1P-1C-O2 45 (0.383) Cm (22:147)
1: Scan ES+
9.00e6
100
%
0
456
458
459
308
353 381 405 438
495
538
571
610 614
677
695
750 779
798
m/z
300 350 400 450 500 550 600 650 700 750 800

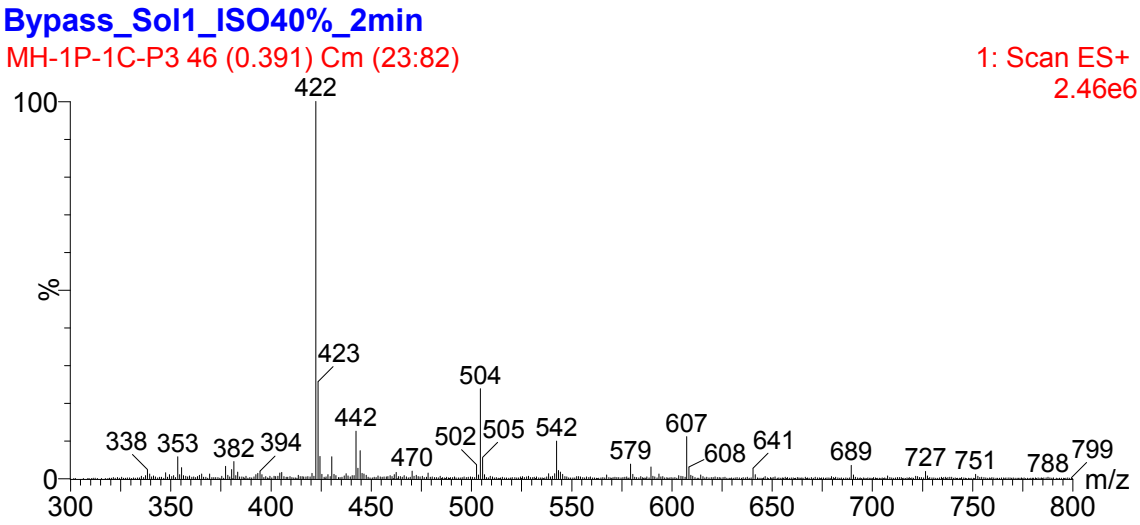
O13 (Green)


Bypass_Sol1_ISO40%_2min
MH-1P-1C-O13
2: Diode Array
Range: 2.174e+1
CO2H
N
N
N
HN
O
MeO
OMe
Exact Mass: 392,20597
Molecular Weight: 392,45600
AU
2.0e+1
1.0e+1
0.0
0.33
0.25 0.50 0.75 1.00 1.25 1.50 1.75 2.00
MH-1P-1C-O13
1: Scan ES+
TIC
9.59e6
100
%
0
0.36
0.38
0.03
0.94 1.11 1.26 1.51 1.63 1.75 1.98
0.25 0.50 0.75 1.00 1.25 1.50 1.75 2.00
MH-1P-1C-O13
1: Scan ES+
393.468
5.81e5
100
%
0
0.34 0.37
0.41
0.31
0.26
0.45
0.48
0.60
Time
0.25 0.50 0.75 1.00 1.25 1.50 1.75 2.00

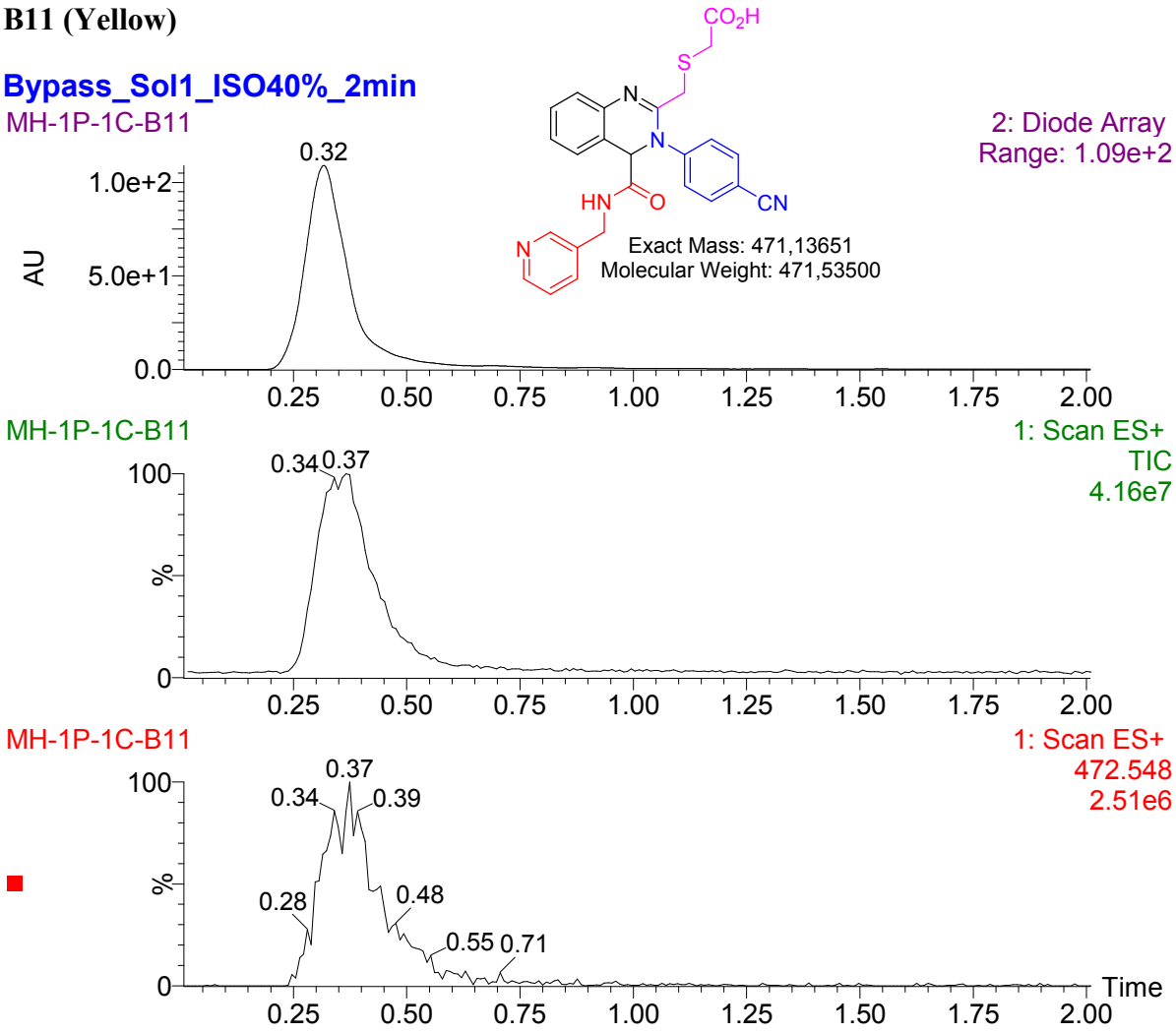

Bypass_Sol1_ISO40%_2min
MH-1P-1C-O13 42 (0.357) Cm (24:70)
1: Scan ES+
1.73e5
100
%
0
393
361
381
353
415
423
479
493
457
539
538
561
577
597
653
668
715
747
768
798
329
339
m/z
300 350 400 450 500 550 600 650 700 750 800

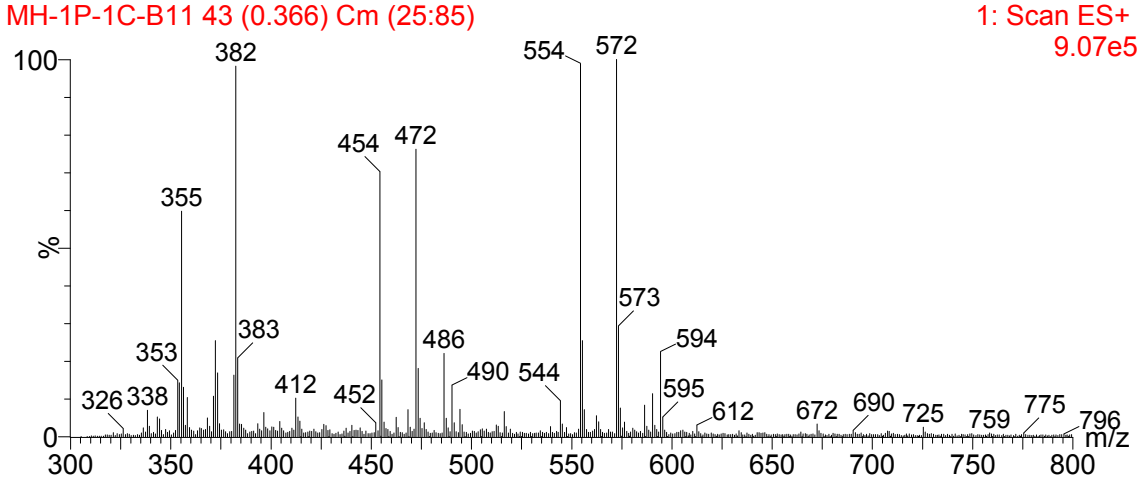
P3 (Green)


Bypass_Sol1_ISO40%_2min
MH-1P-1C-P3
2: Diode Array
Range: 2.614e+1
0.33
AU
2.0e+1
1.0e+1
0.0
Exact Mass: 421,16377
Molecular Weight: 421,45300
MH-1P-1C-P3
1: Scan ES+
TIC
2.71e7
0.35 0.39
100
%
0
MH-1P-1C-P3
1: Scan ES+
422.458
6.42e6
0.35
0.39
0.42
0.29
0.49
Time
0.25 0.50 0.75 1.00 1.25 1.50 1.75 2.00

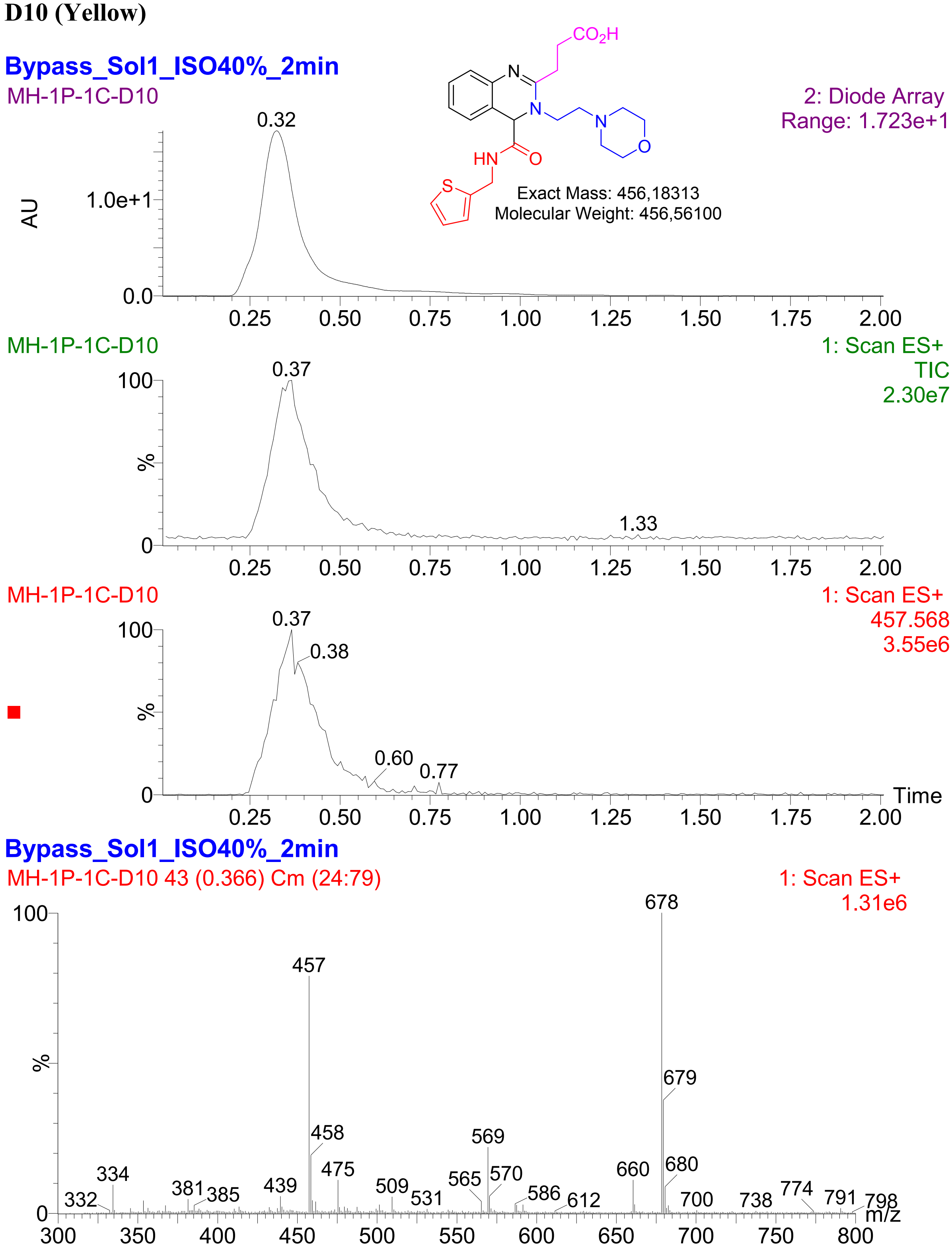

Bypass_Sol1_ISO40%_2min
MH-1P-1C-P3 46 (0.391) Cm (23:82)
1: Scan ES+
2.46e6
100
%
0
422
423
504
442
338 353 382 394
470
502 505 542
579
607
608 641
689
727 751
788 799
m/z
300 350 400 450 500 550 600 650 700 750 800

B11 (Yellow)

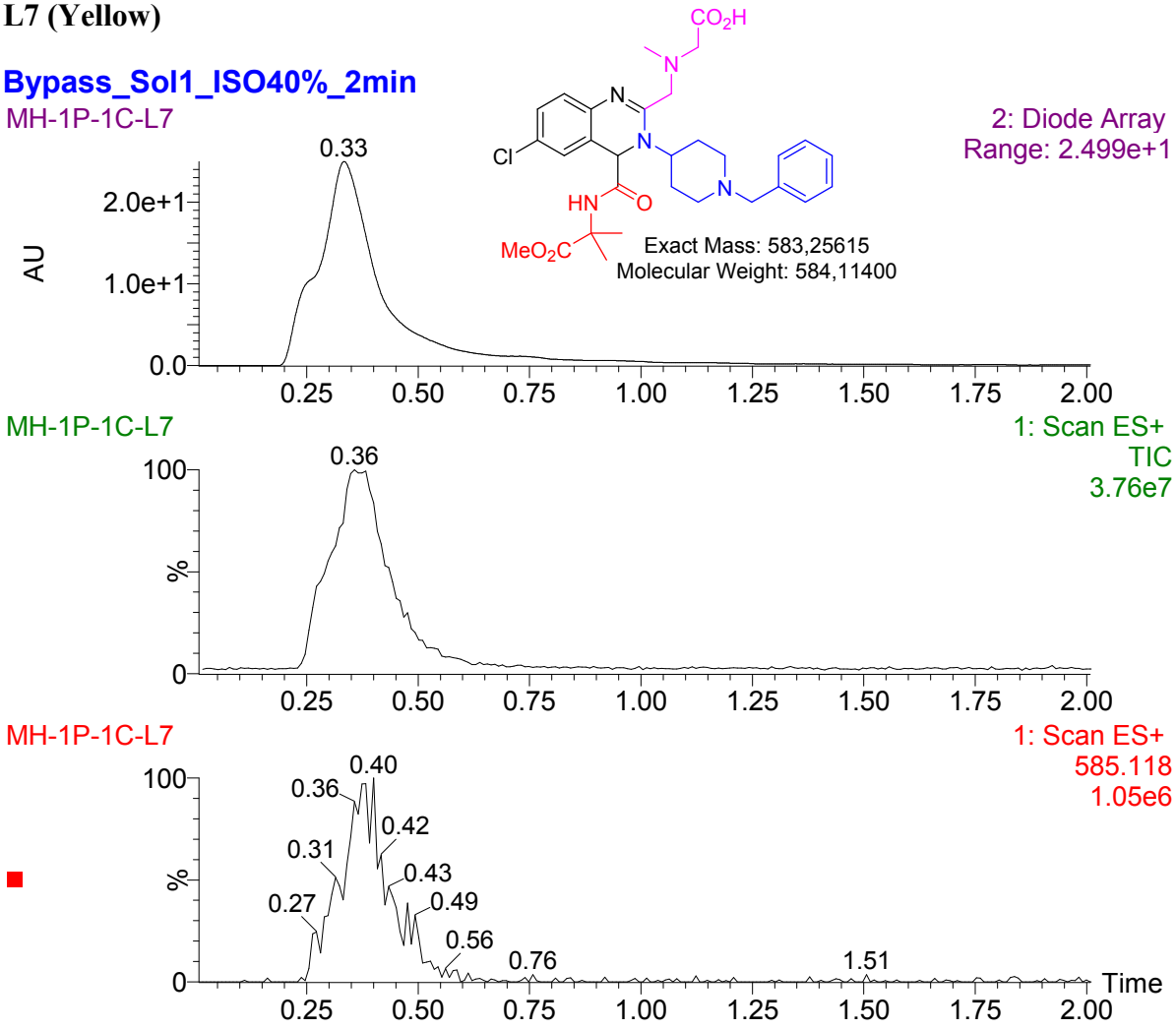

Bypass_Sol1_ISO40%_2min
MH-1P-1C-B11
2: Diode Array
Range: 1.09e+2
0.32
AU
1.0e+2
5.0e+1
0.0
0.25 0.50 0.75 1.00 1.25 1.50 1.75 2.00
CO2H
S
N
N
HN
O
CN
N
Exact Mass: 471,13651
Molecular Weight: 471,53500
MH-1P-1C-B11
1: Scan ES+
TIC
4.16e7
0.34 0.37
100
%
0
0.25 0.50 0.75 1.00 1.25 1.50 1.75 2.00
MH-1P-1C-B11
1: Scan ES+
472.548
2.51e6
0.37
0.34
0.39
0.28
0.48
0.55 0.71
100
%
0
Time
0.25 0.50 0.75 1.00 1.25 1.50 1.75 2.00

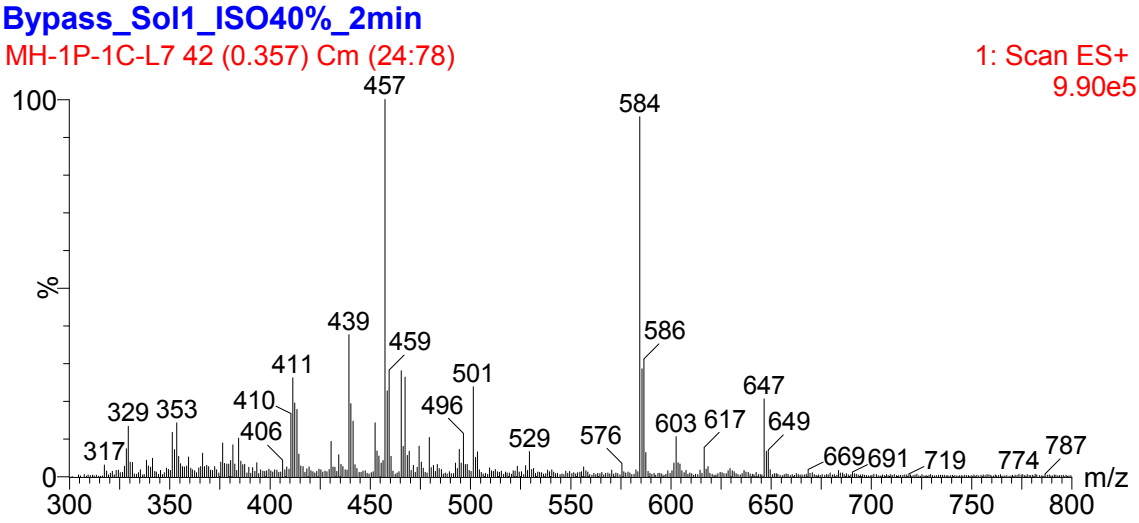

Bypass_Sol1_ISO40%_2min
MH-1P-1C-B11 43 (0.366) Cm (25:85)
1: Scan ES+
9.07e5
100
%
0
382
554
572
454
472
355
573
383
353
486
594
490
544
412
452
595
326
338
612
672
690
725
759
775
796
m/z
300 350 400 450 500 550 600 650 700 750 800

D10 (Yellow)

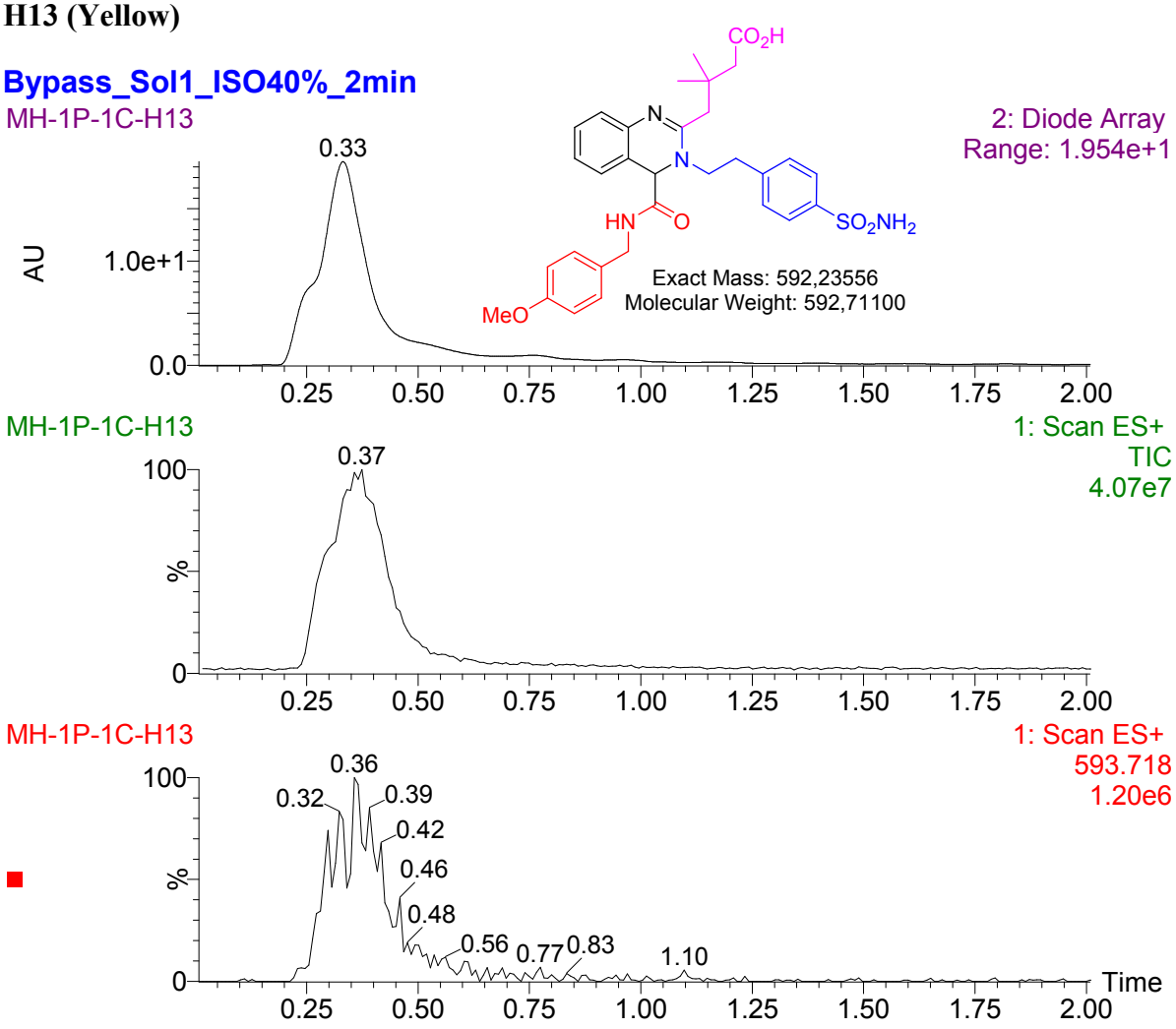

Bypass_Sol1_ISO40%_2min
MH-1P-1C-D10
2: Diode Array
Range: 1.723e+1
CO2H
N
N
N
O
HN
O
S
Exact Mass: 456,18313
Molecular Weight: 456,56100
0.32
AU
1.0e+1
0.0
0.25 0.50 0.75 1.00 1.25 1.50 1.75 2.00
MH-1P-1C-D10
1: Scan ES+
TIC
2.30e7
0.37
100
%
0
1.33
0.25 0.50 0.75 1.00 1.25 1.50 1.75 2.00
MH-1P-1C-D10
1: Scan ES+
457.568
3.55e6
0.37
0.38
100
%
0
0.60
0.77
Time
0.25 0.50 0.75 1.00 1.25 1.50 1.75 2.00
Bypass_Sol1_ISO40%_2min
MH-1P-1C-D10 43 (0.366) Cm (24:79)
1: Scan ES+
1.31e6
100
%
0
678
457
679
458
569
334
475
570
660
680
332
381
385
439
509
531
565
586
612
700
738
774
791
798
m/z
300 350 400 450 500 550 600 650 700 750 800

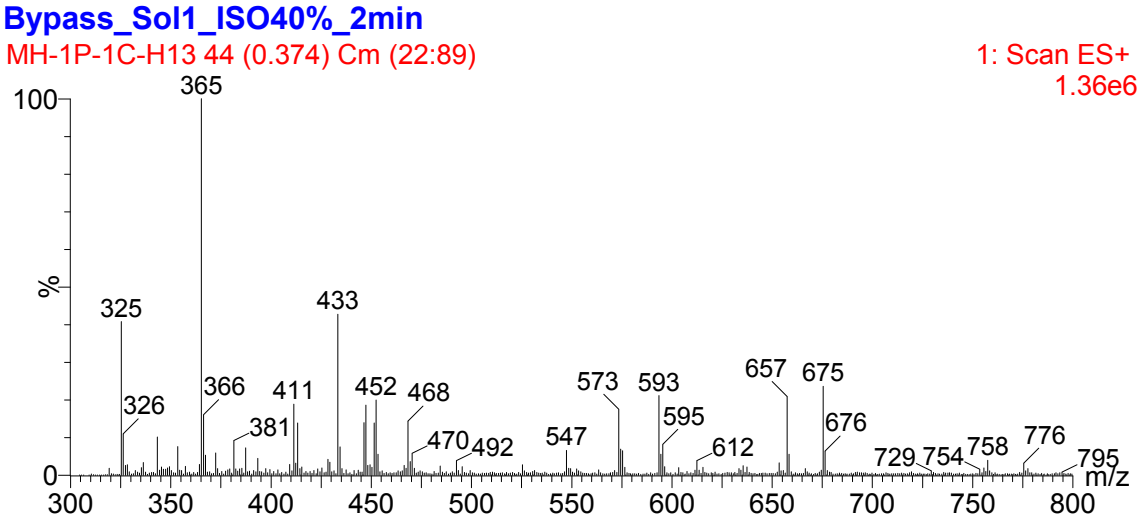
L7 (Yellow)


Bypass_Sol1_ISO40%_2min
MH-1P-1C-L7
2: Diode Array
Range: 2.499e+1
0.33
AU
2.0e+1
1.0e+1
0.0
Exact Mass: 583,25615
Molecular Weight: 584,11400
MH-1P-1C-L7
1: Scan ES+
TIC
3.76e7
0.36
100
%
0
MH-1P-1C-L7
1: Scan ES+
585.118
1.05e6
0.40
0.36
0.42
0.31
0.43
0.27
0.49
0.56
0.76
1.51
Time
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00


Bypass_Sol1_ISO40%_2min
MH-1P-1C-L7 42 (0.357) Cm (24:78)
1: Scan ES+
9.90e5
100
%
0
457
584
439
459
586
411
501
647
329
353
410
496
649
406
603
617
317
529
576
669
691
719
774
787
m/z
300
350
400
450
500
550
600
650
700
750
800

H13 (Yellow)


Bypass_Sol1_ISO40%_2min
MH-1P-1C-H13
2: Diode Array
Range: 1.954e+1
0.33
AU
1.0e+1
0.0
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00
CO2H
N
N
HN
O
SO2NH2
MeO
Exact Mass: 592,23556
Molecular Weight: 592,71100
MH-1P-1C-H13
1: Scan ES+
TIC
4.07e7
0.37
100
%
0
MH-1P-1C-H13
1: Scan ES+
593.718
1.20e6
0.36
0.32
0.39
0.42
0.46
0.48
0.56
0.77
0.83
1.10
Time


Bypass_Sol1_ISO40%_2min
MH-1P-1C-H13 44 (0.374) Cm (22:89)
1: Scan ES+
1.36e6
100
%
0
365
325
326
366
381
411
433
452
468
470
492
547
573
593
595
612
657
675
676
729
754
758
776
795
m/z
300
350
400
450
500
550
600
650
700
750
800

4. Heat plot

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
A	1	1	1	1	0	1	1	0	1	1	0,1	1	0	0	0	0,1	0,1	0	1	1	1	1	0	1
B	1	0,2	1	1	0,1	1	1	0,3	0	0,2	0,8	0	1	1	0,1	0	0	0	0	0,7	0	1	0	0
C	1	0	0,5	0,2	1	0,7	1	1	0,2	1	1	0,3	0	0	0,7	1	0,1	1	0,1	0	0	0,1	0,6	1
D	0	1	0	0,5	0,1	0,7	1	1	1	0,8	1	1	1	0,1	0,2	1	1	0,2	1	0	0,1	0,3	0	0,3
E	0	1	0,7	1	0	1	1	1	1	0,2	1	1	1	0,7	1	0,4	0,2	0,1	0,3	0,1	1	1	0,2	0
F	1	0,1	1	0	0	1	0,9	0,3	0,3	1	0	0	0	0,3	0	1	1	1	0	0	1	0,5	0	0,5
G	0	1	0	0,1	0,3	0	1	0,3	0,1	1	1	0	0,1	0,4	0,2	0,1	0,1	0	0,3	0,3	1	1	1	0
H	0,1	0	1	0	0	1	0,4	0	0,1	1	1	0	0,2	0,2	0,7	0	1	1	1	1	1	0	0	0
I	0,1	0,6	1	0	0	0	1	0	0,1	0,1	0	0,1	0,4	1	0,3	0,3	1	0	0,6	0,1	0	0	0	1
J	0	0,7	0	0,6	0,9	1	1	1	0,1	0,1	0,1	0,4	1	0,4	0,1	0,6	0	1	1	0,2	1	0	1	0
K	0	0	0,4	0,4	1	1	0	1	0	0	0	0,2	1	1	0,1	1	1	1	0,5	1	0,1	0	1	0,5
L	0	0	0	0	0	1	0,7	1	0,1	1	1	0,3	0,1	0	0,7	0	0,1	0	1	0	1	1	0,3	0,1
M	1	1	0,6	0,1	0	0,1	0,2	0	0	1	1	0	0	0	1	1	1	0	1	1	1	0,4	0,1	1
N	0	0	1	0,2	1	0,1	0,7	1	0	1	1	0,5	0,2	0,3	0,1	0,4	0	0	0	0	1	1	0,1	0
O	0,3	1	0,9	1	1	0,4	1	1	1	1	0	0,1	1	0	1	1	1	0,1	1	0,1	1	0,4	1	0,3
P	1	0,1	1	0,1	1	0,1	0,1	0	1	1	0	0,5	1	0,1	0	0,1	1	0,6	0	1	0,1	0	1	0,1

Major product formation Medium product formation No product formation Resynthesized

5. Statistical reaction analysis

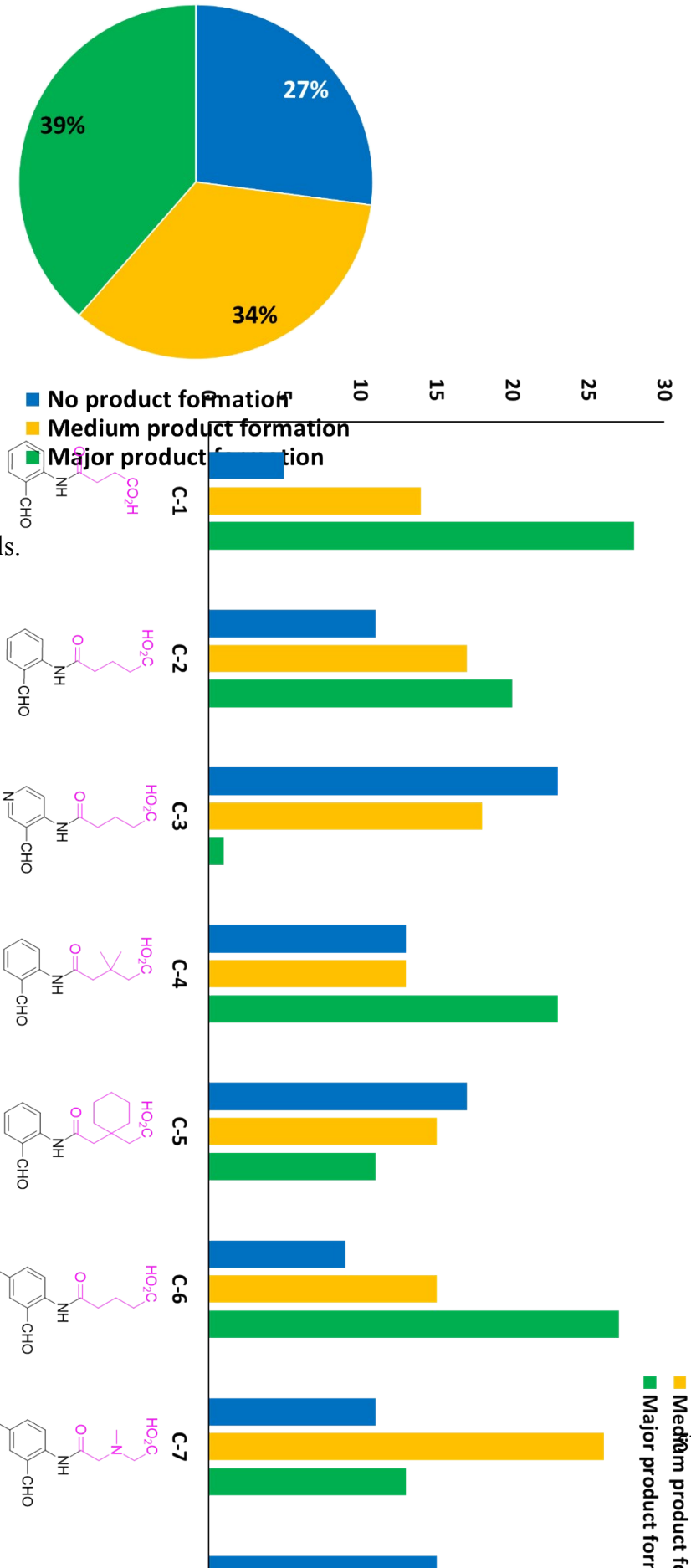


Fig. S3. QC results of 384 wells.

Fig. S4. Performance of α-formyl-ω-carboxylic acids in the 384-well plate.

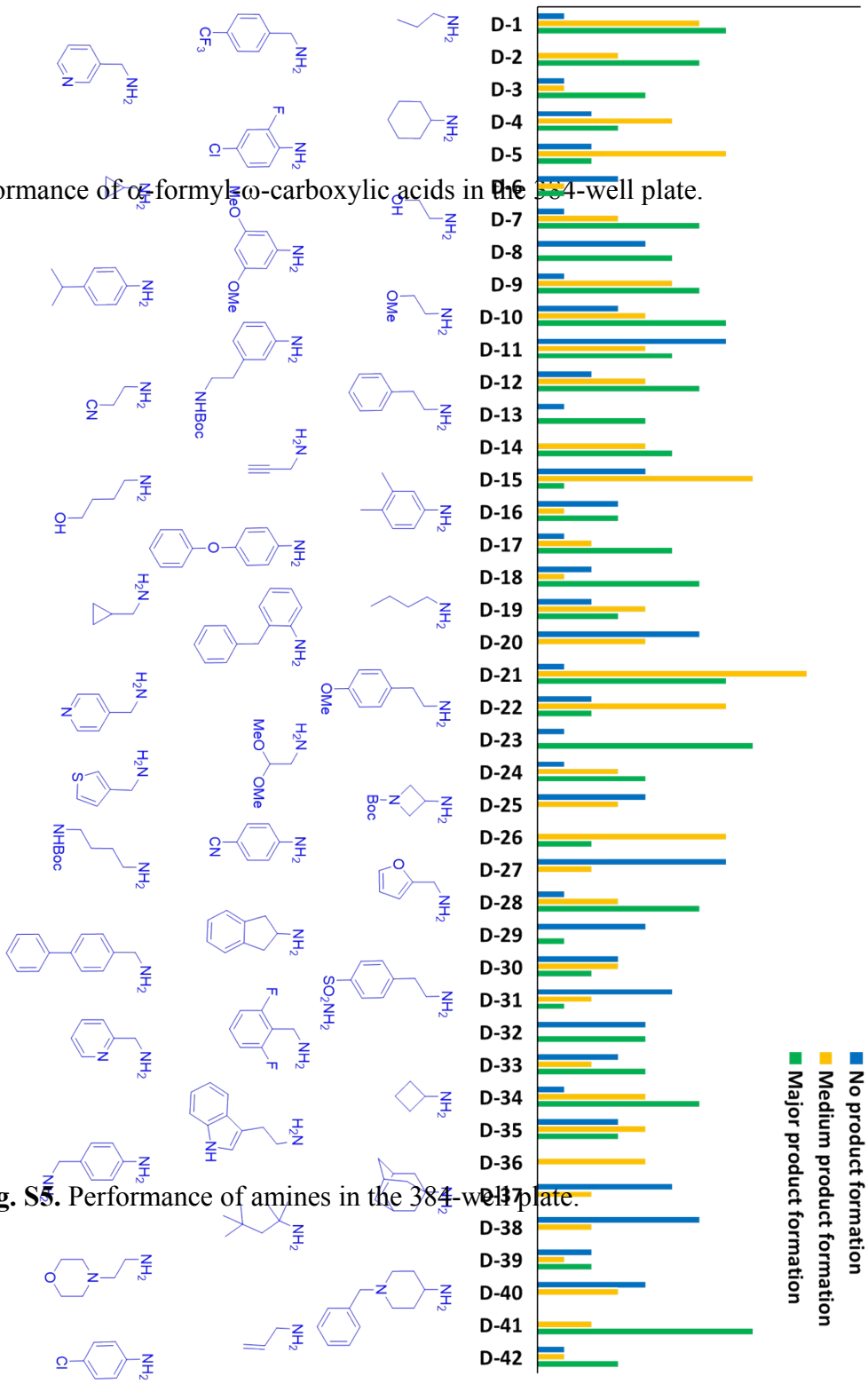


Fig. S5. Performance of amines in the 384-well plate.

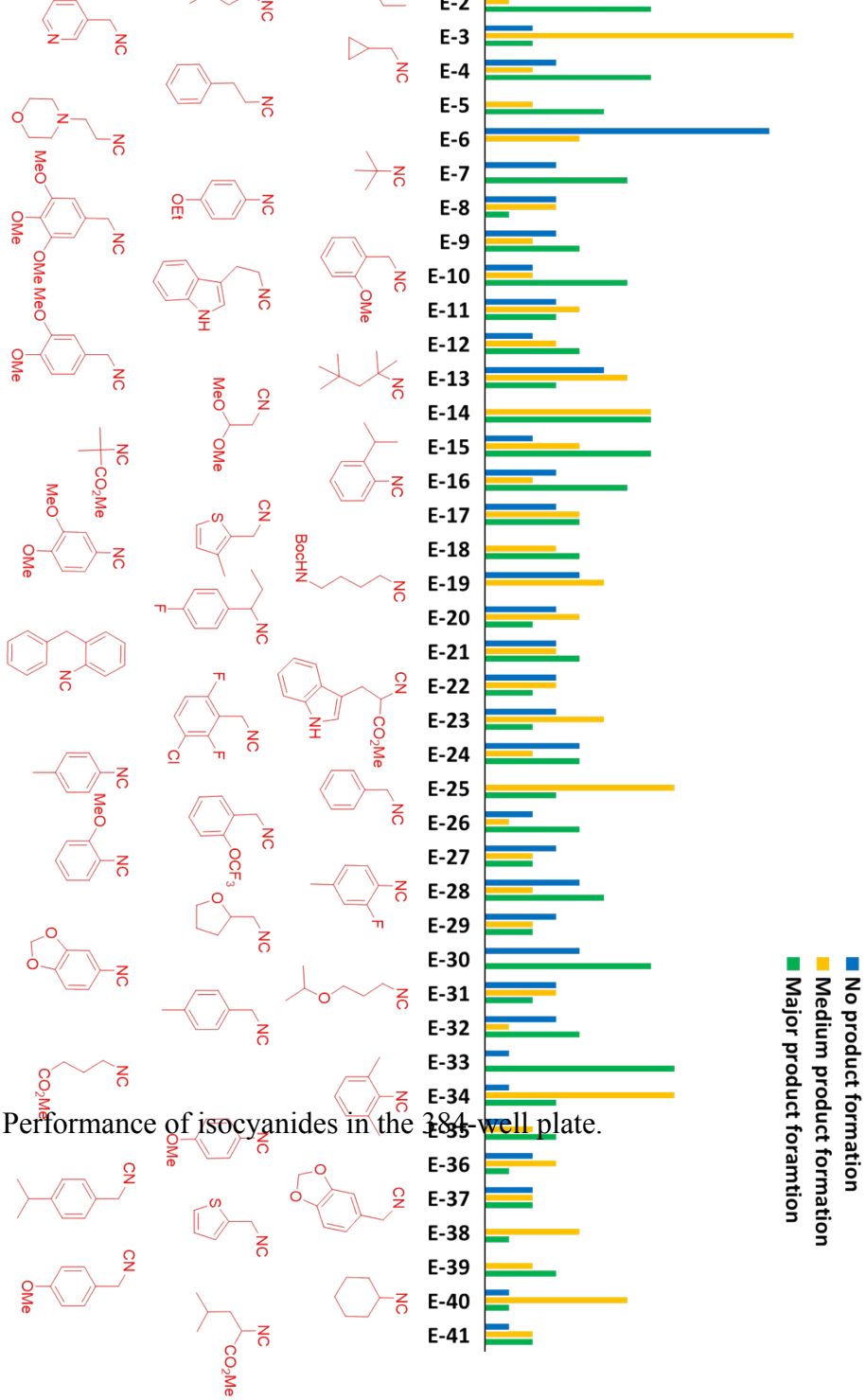
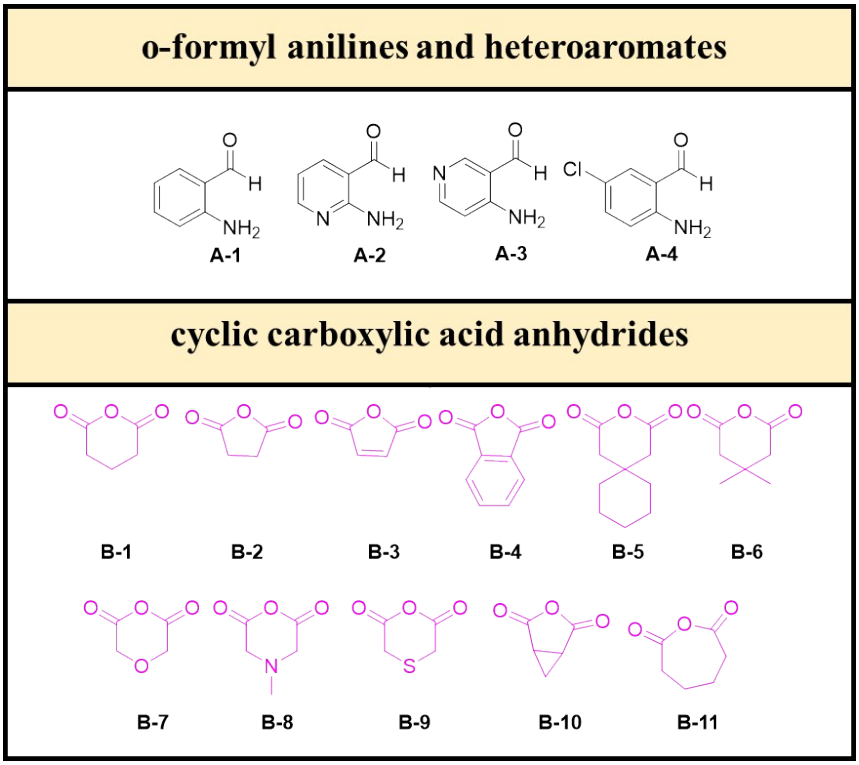


Fig. S6. Performance of isocyanides in the 384-well plate.

6. General experimental data for the mmol scale synthesis

6.1. Structures of the building blocks



6.2. mmol scale synthesis procedures

Procedure A: General procedure for α-formyl-ω-carboxylic acids (C): To the stirred solution of cyclic anhydride (**B**, 6 mmol) in 1,4-dioxane (36 mL), 2-aminobenzaldehyde (**A**, 6.0 mmol) was added. The reaction mixture was stirred and refluxed for 18 h (overnight). The solvent was removed under reduced pressure. The crude reaction mixture was purified by flash column chromatography (230-400 mesh size silica gel) eluting with CH_2Cl_2–MeOH (20:1) mixture of solvents (**C-4** was eluted with petroleum ether: ethyl acetate (3:1)) to afford the corresponding product **(C)**.

Procedure B: General procedure for 2,3,4-trisubstituted quinazolines (mmol scale): To the stirred solution of α-formyl-ω-carboxylic acids (**C**, 1.0 mmol) in trifluoroethanol (5 ml), amine (**D**) (1.1 mmol) and isocyanide (**E**) (1.1 mmol) were added. The corresponding reaction mixture was stirred at room temperature for 18 h. The solvent was removed under reduced pressure, the residue obtained was purified by flash column chromatography (230-400 mesh size silica gel) eluting with CH_2Cl_2–MeOH (5:1) mixture of solvents to afford the corresponding final product.

C-1: 4-((2-formylphenyl)amino)-4-oxobutanoic acid:

The product was synthesized according to procedure **A** in 6 mmol scale as off white solid (1142 mg, 86% yield), M.P.= 123 – 125 °C. ^{1}H NMR (500 MHz, $CDCl_3$) δ 11.20 (s, 1H), 9.89 (s, 1H), 8.68 (d, *J* = 8.4 Hz, 1H), 7.65 (d, *J* = 7.6 Hz, 1H), 7.58 (t, *J* = 7.8 Hz, 1H), 7.21 (t, *J* = 7.8 Hz, 1H), 2.78 (m, 4H); ^{13}C NMR (126 MHz, $CDCl_3$) δ 195.6, 178.1, 170.8, 140.7, 136.3, 136.1, 123.1, 121.6, 120.0, 32.3, 28.9; HRMS calcd for $C_{11}H_{12}NO_4$ $[M+H]^+$: 222.0761, found $[M+H]^+$: 222.0760.

C-2: 5-((2-formylphenyl)amino)-5-oxopentanoic acid:

The product was synthesized according to procedure **A** in 6 mmol scale as off white solid (1789 mg, 95% yield), M.P.= 105 – 107 °C. ^{1}H NMR (500 MHz, $CDCl_3$) δ 11.16 (s, 1H), 9.91 (s, 1H), 8.73 (d, *J* = 8.3 Hz, 1H), 7.67 (d, *J* = 7.5 Hz, 1H), 7.61 (t, *J* = 7.8 Hz, 1H), 7.23 (t, *J* = 7.5 Hz, 1H), 2.57 (t, *J* = 7.8 Hz, 2H), 2.50 (t, *J* = 6.4 Hz, 2H), 2.12 – 2.06 (m, 2H); ^{13}C NMR (126 MHz, $CDCl_3$) δ 195.6, 178.2, 171.7, 140.8, 136.2, 136.1, 123.0, 121.5, 119.9, 37.0, 32.9, 20.1; HRMS calcd for $C_{12}H_{14}NO_4$ $[M+H]^+$: 236.0917, found $[M+H]^+$: 236.0917.

C-3: 5-((3-formylpyridin-4-yl)amino)-5-oxopentanoic acid:

The product was synthesized according to procedure **A** in 6 mmol scale as off white solid (665 mg, 47% yield), M.P.= 75 – 76 °C. ^{1}H NMR (500 MHz, DMSO-d_6) δ 12.11 (s, 1H), 10.95 (s, 1H), 10.04 (s, 1H), 8.95 (s, 1H), 8.66 (d, *J* = 5.8 Hz, 1H), 8.30 (d, *J* = 5.8 Hz, 1H), 2.54 (t, *J* = 7.4 Hz, 2H), 2.31 (t, *J* = 7.4 Hz, 2H), 1.87 – 1.80 (m, 2H); ^{13}C NMR (126 MHz, DMSO-d_6) δ 195.0, 174.0, 172.6, 156.2, 155.2, 145.6, 117.9, 113.3, 36.2, 32.6, 19.9; HRMS calcd for $C_{11}H_{13}N_2O_4$ $[M+H]^+$: 237.0870, found $[M+H]^+$: 237.0868.

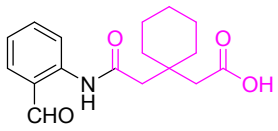
C-4: 5-((2-formylphenyl)amino)-3,3-dimethyl-5-oxopentanoic acid:

The product was synthesized according to procedure **A** in 6 mmol scale as light yellow solid (947 mg, 60% yield), M.P.= 117 – 120 °C. ^{1}H NMR (500 MHz, $CDCl_3$) δ 11.29 (s, 1H), 9.92 (s, 1H), 8.72 (d, *J* = 8.4 Hz, 1H), 7.70 (dd, *J* = 7.6, 1.7 Hz, 1H), 7.66 – 7.60 (m, 1H), 7.29 (d, *J* = 7.6 Hz, 1H), 2.56 (s, 2H), 2.51 (s, 2H), 1.20 (s, 6H); ^{13}C NMR (126 MHz, $CDCl_3$) δ 195.6, 172.2, 140.0, 136.2, 136.1, 123.6, 121.9,

120.2, 49.0, 45.7, 33.6, 32.3, 28.5, 27.8; HRMS calcd for $C_{14}H_{18}NO_4$ $[M+H]^+$: 264.1230, found $[M+H]^+$: 264.1230.

C-5: 2-(1-(2-((2-formylphenyl)amino)-2-oxoethyl)cyclohexyl)acetic acid:

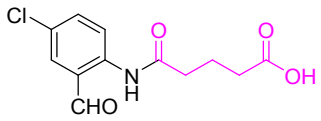
The product was synthesized according to procedure **A** in 6 mmol scale as light yellow solid (1273



mg, 70% yield), M.P.= 121 – 124 °C. ^{1}H NMR (500 MHz, $CDCl_3$) δ 11.42 (s, 1H), 9.93 (s, 1H), 8.71 (d, *J* = 8.4 Hz, 1H), 7.72 (dd, *J* = 7.8, 1.0 Hz, 1H), 7.65 (td, *J* = 7.8, 1.1 Hz, 1H), 7.31 (t, *J* = 7.5 Hz, 1H), 2.63 (s, 2H), 2.57 (s, 2H), 1.60 – 1.51 (m, 8H), 1.50 – 1.44 (m, 2H); ^{13}C NMR (126 MHz, $CDCl_3$) δ 195.6, 173.5, 173.1, 139.6, 136.2, 136.1, 124.0, 122.0, 120.3, 46.3, 42.9, 36.6, 25.6, 21.3; HRMS calcd for $C_{17}H_{22}NO_4$ $[M+H]^+$: 304.1543, found $[M+H]^+$: 304.1541.

C-6: 5-((4-chloro-2-formylphenyl)amino)-5-oxopentanoic acid:

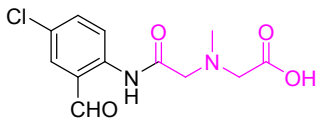
The product was synthesized according to procedure **A** in 6 mmol scale as light pink solid (904



mg, 56% yield), M.P.= 136 – 138 °C. ^{1}H NMR (500 MHz, $CDCl_3$) δ 11.05 (s, 1H), 9.86 (s, 1H), 8.72 (d, *J* = 9.0 Hz, 1H), 7.63 (d, *J* = 2.5 Hz, 1H), 7.55 (dd, *J* = 9.0, 2.5 Hz, 1H), 2.56 (t, *J* = 7.2 Hz, 2H), 2.50 (t, *J* = 7.2 Hz, 2H), 2.12 – 2.04 (m, 2H); ^{13}C NMR (126 MHz, $CDCl_3$) δ 194.4, 178.5, 171.7, 139.2, 136.0, 135.0, 128.0, 122.5, 121.6, 36.9, 32.9, 20.0; HRMS calcd for $C_{12}H_{13}ClNO_4$ $[M+H]^+$: 270.0528, found $[M+H]^+$: 270.0527.

C-7: N-(2-((4-chloro-2-formylphenyl)amino)-2-oxoethyl)-N-methylglycine:

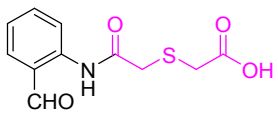
The product was synthesized according to procedure **A** in 6 mmol scale as light pink solid (1278



mg, 75% yield), M.P.= 150 – 153 °C. ^{1}H NMR (500 MHz, CD_3OD) δ 9.93 (s, 1H), 8.58 (d, *J* = 9.0 Hz, 1H), 7.88 (d, *J* = 2.3 Hz, 1H), 7.64 (dd, *J* = 9.0, 2.3 Hz, 1H), 3.71 (s, 2H), 3.62 (s, 2H), 2.69 (s, 3H); ^{13}C NMR (126 MHz, DMSO-d_6) δ 194.4, 172.0, 170.9, 138.0, 135.0, 134.7, 126.8, 124.1, 121.3, 60.4, 57.2, 42.3; HRMS calcd for $C_{12}H_{14}ClNO_4$ $[M+H]^+$: 285.0637, found $[M+H]^+$: 285.0636.

C-8: 2-((2-((2-formylphenyl)amino)-2-oxoethyl)thio)acetic acid:

The product was synthesized according to procedure **A** in 6 mmol scale as light yellow solid (1351



mg, 89% yield), M.P.= 119 – 121 °C. ^{1}H NMR (500 MHz, $CDCl_3$) δ 11.73 (s, 1H), 9.93 (s, 1H), 8.71 (d, *J* = 8.4 Hz, 1H), 7.71 (dd, *J* = 7.6, 1.5 Hz,

1H), 7.66 – 7.60 (m, 1H), 7.28 (dd, J = 7.6, 1.5 Hz, 1H), 3.59 (s, 2H), 3.41 (s, 2H); ^{13}C NMR (126 MHz, $CDCl_3$) δ 195.4, 174.2, 168.5, 139.9, 136.1, 136.0, 123.7, 122.1, 120.1, 37.4, 33.7; HRMS calcd for $C_{11}H_{12}NO_4S$ $[M+H]^+$: 254.0482, found $[M+H]^+$: 254.0481.

C-9: 2-(2-((2-formylphenyl)amino)-2-oxoethoxy)acetic acid:

The product was synthesized according to procedure **A** in 6 mmol scale as light pink solid (1289 mg, 91% yield), M.P.= 161 – 162 °C. ^{1}H NMR (500 MHz, DMSO-d_6) δ 12.86 (s, 1H), 11.63 (s, 1H), 9.98 (s, 1H), 8.56 (d, J = 8.2 Hz, 1H), 7.94 (d, J = 8.2 Hz, 1H), 7.70 (t, J = 7.5 Hz, 1H), 7.34 (t, J = 7.5 Hz, 1H), 4.28 (s, 2H), 4.22 (s, 2H); ^{13}C NMR (126 MHz, DMSO-d_6) δ 195.8, 171.2, 169.1, 139.0, 135.8, 123.6, 122.8, 119.5, 70.3, 67.9; HRMS calcd for $C_{11}H_{12}NO_5$ $[M+H]^+$: 238.0710, found $[M+H]^+$: 238.0709.

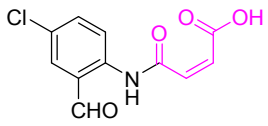
C-10: 5-((3-formylpyridin-2-yl)amino)-5-oxopentanoic acid:

The product was synthesized according to procedure **A** in 6 mmol scale as yellow solid (736 mg, 52% yield), M.P.= 182 – 183 °C. ^{1}H NMR (500 MHz, DMSO-d_6) δ 12.11 (s, 1H), 10.83 (s, 1H), 9.73 (s, 1H), 8.62 (dd, J = 4.8, 1.9 Hz, 1H), 8.12 (dd, J = 7.7, 1.9 Hz, 1H), 7.39 (dd, J = 7.7, 4.8 Hz, 1H), 2.53 (t, J = 7.4 Hz, 2H), 2.28 (t, J = 7.4 Hz, 2H), 1.85 – 1.77 (m, 2H); ^{13}C NMR (126 MHz, DMSO-d_6) δ 189.0, 174.1, 173.0, 152.7, 150.6, 136.8, 123.1, 121.1, 34.7, 32.9, 20.1; HRMS calcd for $C_{11}H_{13}N_2O_4$ $[M+H]^+$: 237.0870, found $[M+H]^+$: 237.0868.

C-11: (Z)-4-((2-formylphenyl)amino)-4-oxobut-2-enoic acid:

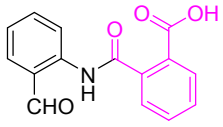
The product was synthesized according to procedure **A** in 6 mmol scale as light yellow solid (1116 mg, 85% yield), M.P.= 159 – 161 °C. ^{1}H NMR (500 MHz, $CDCl_3$) δ 11.98 (s, 1H), 9.97 (s, 1H), 8.72 (d, J = 8.4 Hz, 1H), 7.80 (d, J = 7.6 Hz, 1H), 7.73 (t, J = 7.9 Hz, 1H), 7.44 (t, J = 7.6 Hz, 1H), 6.53 (d, J = 12.8 Hz, 1H), 6.48 (d, J = 12.8 Hz, 1H); ^{13}C NMR (126 MHz, $CDCl_3$) δ 196.0, 165.4, 164.0, 138.7, 138.2, 136.6, 136.4, 131.6, 125.7, 122.5, 121.0; HRMS calcd for $C_{11}H_{10}NO_4$ $[M+H]^+$: 220.0604, found $[M+H]^+$: 220.0604.

C-12: (Z)-4-((4-chloro-2-formylphenyl)amino)-4-oxobut-2-enoic acid:

The product was synthesized according to procedure **A** in 6 mmol scale as yellow solid (1063 mg, 

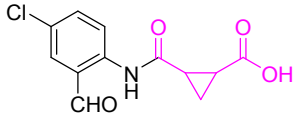
70% yield), M.P.= 157 – 159 °C. ^{1}H NMR (500 MHz, DMSO-d_6) δ 12.96 (s, 1H), 10.89 (s, 1H), 9.97 (s, 1H), 7.95 (d, J = 8.8 Hz, 1H), 7.86 (d, J = 2.5 Hz, 1H), 7.75 (dd, J = 8.8, 2.5 Hz, 1H), 6.63 (d, J = 12.0 Hz, 1H), 6.33 (d, J = 12.0 Hz, 1H); ^{13}C NMR (126 MHz, DMSO-d_6) δ 191.9, 166.5, 164.7, 137.8, 134.5, 133.1, 130.4, 129.3, 128.8, 127.3, 124.5; HRMS calcd for $C_{11}H_9ClNO_4$ $[M+H]^+$: 254.0215, found $[M+H]^+$: 254.0214.

C-13: 2-((2-formylphenyl)carbamoyl)benzoic acid:

The product was synthesized according to procedure **A** in 6 mmol scale as off white solid (1340 

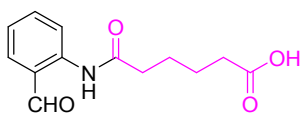
mg, 83% yield), M.P.= 151 – 152 °C. ^{1}H NMR (500 MHz, DMSO-d_6) δ 13.25 (s, 1H), 11.06 (s, 1H), 10.08 (s, 1H), 8.12 (d, J = 8.1 Hz, 1H), 7.93 – 7.88 (m, 2H), 7.76 – 7.61 (m, 4H), 7.38 (t, J = 7.5, 1H); ^{13}C NMR (126 MHz, DMSO-d_6) δ 194.0, 168.1, 167.5, 139.9, 138.0, 135.3, 132.1, 130.2, 129.8, 127.6, 125.4, 124.5, 122.0; HRMS calcd for $C_{15}H_{12}NO_4$ $[M+H]^+$: 270.0761, found $[M+H]^+$: 270.0760.

C-14: 2-((4-chloro-2-formylphenyl)carbamoyl)cyclopropane-1-carboxylic acid:

The product was synthesized according to procedure **A** in 6 mmol scale as light pink solid (1281 

mg, 80% yield), M.P.= 172 – 176 °C. ^{1}H NMR (500 MHz, CD_3OD) δ 9.89 (s, 1H), 8.39 (d, J = 8.9 Hz, 1H), 7.81 (d, J = 2.6 Hz, 1H), 7.56 (dd, J = 8.9, 2.6 Hz, 1H), 2.35 (td, J = 8.6, 7.0 Hz, 1H), 2.16 (td, J = 9.1, 8.6, 6.1 Hz, 1H), 1.68 – 1.63 (m, 1H), 1.36 (td, J = 8.6, 4.8 Hz, 1H); ^{13}C NMR (126 MHz, CD_3OD) δ 194.1, 172.8, 169.0, 138.4, 134.8, 133.7, 128.4, 124.4, 121.9, 25.2, 20.7, 10.9; HRMS calcd for $C_{12}H_{11}ClNO_4$ $[M+H]^+$: 268.0371, found $[M+H]^+$: 268.0371.

C-15: 6-((2-formylphenyl)amino)-6-oxohexanoic acid:

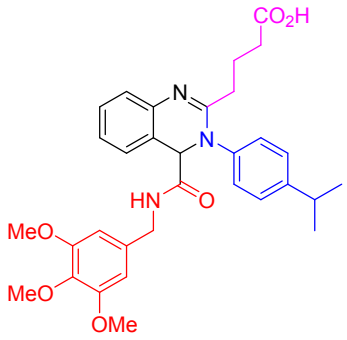
The product was synthesized according to procedure **A** in 6 mmol scale as off white solid (672 

mg, 45% yield), M.P.= 98 – 100 °C. ^{1}H NMR (500 MHz, $CDCl_3$) δ 11.16 (s, 1H), 9.91 (s, 1H), 8.73 (d, J = 8.5 Hz, 1H), 7.67 (d, J = 8.5 Hz, 1H), 7.61 (t, J = 7.9 Hz, 1H), 7.22 (t, J = 7.5 Hz, 1H), 2.49 (t, J = 7.3 Hz, 2H), 2.42 (t, J = 7.2 Hz, 2H), 1.86 – 1.78 (m, 2H), 1.78 – 1.70 (m, 2H); ^{13}C NMR (126 MHz,

$CDCl_3$) δ 195.6, 178.9, 172.2, 140.8, 136.2, 136.0, 122.9, 121.5, 119.9, 37.9, 33.6, 24.6, 24.1; HRMS calcd for $C_{13}H_{16}NO_4$ $[M+H]^+$: 250.1074, found $[M+H]^+$: 250.1074.

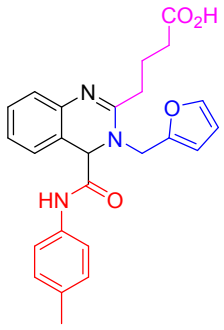
A24:4-(3-(4-isopropylphenyl)-4-((3,4,5-trimethoxybenzyl)carbamoyl)-3,4-dihydroquinazolin-2-yl)butanoic acid:

The product was synthesized according to procedure **B** in 1 mmol scale as off white solid (447 mg, 80% yield), M.P.= 183 – 185 °C. I.R. (thin film): 2919, 2358, 1682, 1538, 1329, 756 cm^{-1}. ^1H NMR (500 MHz, $CDCl_3$) δ 9.54 (s, 1H), 7.71 (d, *J* = 7.2 Hz, 1H), 7.26 (s, 1H), 7.23 – 7.07 (m, 6H), 7.03 (t, *J* = 7.0 Hz, 1H), 6.35 (s, 2H), 5.82 (s, 1H), 4.33 (dd, *J* = 14.1, 4.7 Hz, 1H), 4.17 (d, *J* = 9.9 Hz, 1H), 3.73 (s, 3H), 3.61 (s, 6H), 2.92 – 2.81 (m, 1H), 2.58-2.42 (m, 2H), 2.42 – 2.19 (m, 2H), 1.95 – 1.64 (m, 2H), 1.19 (d, *J* = 6.8 Hz, 6H); ^{13}C NMR (126 MHz, $CDCl_3$) δ 178.1, 168.4, 162.3, 153.0, 150.9, 137.3, 136.6, 134.1, 131.7, 129.6, 128.3, 127.1, 126.6, 118.9, 118.4, 104.1, 66.1, 60.7, 56.0, 55.9, 43.3, 35.1, 33.7, 30.8, 23.6, 22.8; HRMS calcd for $C_{32}H_{38}N_3O_6$ $[M+H]^+$: 560.2755, found $[M+H]^+$: 560.2751.



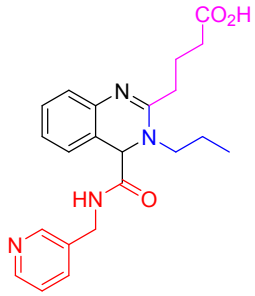
B13:4-(3-(furan-2-ylmethyl)-4-(p-tolylcarbamoyl)-3,4-dihydroquinazolin-2-yl)butanoic acid:

The product was synthesized according to procedure **B** in 1 mmol scale as off white solid (380 mg, 88% yield), M.P.= 165 – 166 °C. I.R. (thin film): 2918, 2324, 1682, 1567, 1497, 755 cm^{-1}. ^1H NMR (500 MHz, CD_3OD) δ 7.56 – 7.46 (m, 2H), 7.39 (s, 1H), 7.33 (d, *J* = 8.0 Hz, 2H), 7.29 (t, *J* = 7.6 Hz, 1H), 7.20 (t, *J* = 7.6 Hz, 1H), 7.11 (d, *J* = 8.0 Hz, 1H), 7.03 (d, *J* = 8.0 Hz, 2H), 6.61 (d, *J* = 3.3 Hz, 1H), 6.27 (m, 1H), 5.71 (s, 1H), 5.00 (d, *J* = 16.0 Hz, 1H), 4.77 (d, *J* = 16.0 Hz, 1H), 3.17 – 3.07 (m, 1H), 3.07 – 2.98 (m, 1H), 2.49 – 2.43 (m, 2H), 2.23 (s, 3H), 2.18-2.08 (m, 2H); ^{13}C NMR (126 MHz, CD_3OD + 3 drops of $CDCl_3$) δ 179.3, 166.3, 163.8, 146.3, 144.8, 135.4, 135.0, 131.3, 130.7, 129.8, 128.2, 126.8, 120.9, 118.7, 117.8, 112.9, 111.3, 63.3, 48.0, 36.1, 31.3, 23.9, 21.0; HRMS calcd for $C_{25}H_{26}N_3O_4$ $[M+H]^+$: 432.1918, found $[M+H]^+$: 432.1914.



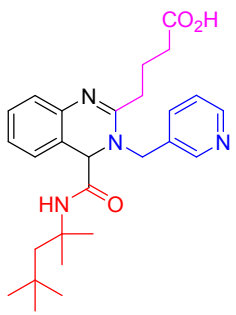
B20:4-(3-propyl-4-((pyridin-3-ylmethyl)carbamoyl)-3,4-dihydroquinazolin-2-yl)butanoic acid:

The product was synthesized according to procedure **B** in 1 mmol scale as off white solid (320 mg, 81% yield), M.P.= 71 – 73 °C. I.R. (thin film): 3213, 2935, 2359, 1681, 1513, 1386, 709 cm^{-1}. ^1H NMR (500 MHz, $CDCl_3$) δ 10.50 (s, 1H), 8.46 (s, 1H), 8.36 (d, *J* = 4.3 Hz, 1H), 7.54 (d, *J* = 7.9 Hz, 1H), 7.50 (d, *J* = 7.5 Hz, 1H), 7.16 – 7.07 (m, 2H), 6.99 (t, *J* = 7.5 Hz, 1H), 6.87 (s, 1H), 5.83 (s, 1H), 4.51-4.26 (m, 2H), 3.64 – 3.52 (m, 1H), 3.24-3.12 (m, 1H), 2.89 – 2.74 (m, 2H), 2.56 – 2.43 (m, 2H), 2.17 – 1.92 (m, 2H), 1.80 – 1.54 (m, 2H), 0.86 (t, *J* = 7.0 Hz, 3H); ^{13}C NMR (126 MHz, $CDCl_3$) δ 178.4, 169.0, 161.9, 149.0, 148.3, 135.5, 134.2, 131.3, 129.3, 126.8, 126.3, 123.3, 117.9, 62.2, 52.4, 41.0, 35.9, 30.0, 22.9, 20.9, 10.9; HRMS calcd for $C_{22}H_{27}N_4O_3$ $[M+H]^+$: 395.2078, found $[M+H]^+$: 395.2074.



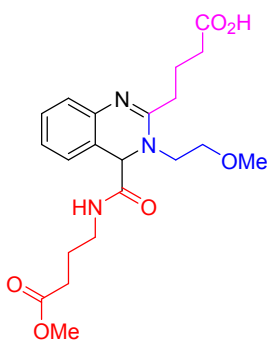
C1:4-(3-(pyridin-3-ylmethyl)-4-((2,4,4-trimethylpentan-2-yl)carbamoyl)-3,4-dihydroquinazolin-2-yl)butanoic acid:

The product was synthesized according to procedure **B** in 1 mmol scale as off white solid (187 mg, 40% yield), M.P.= 69 – 70 °C. I.R. (thin film): 3217, 2950, 2357, 1681, 1544, 1223, 708 cm^{-1}. ^1H NMR (500 MHz, $CDCl_3$) δ 10.27 (brs, 1H), 8.50 (s, 1H), 8.46 (d, *J* = 4.8 Hz, 1H), 7.60 (d, *J* = 7.8 Hz, 1H), 7.24 – 7.16 (m, 3H), 7.12 (d, *J* = 7.8 Hz, 1H), 7.09 – 7.01 (m, 1H), 6.39 (s, 1H), 5.14 (d, *J* = 16.4 Hz, 1H), 5.06 (s, 1H), 4.65 (d, *J* = 16.4 Hz, 1H), 2.87 – 2.76 (m, 1H), 2.76 – 2.65 (m, 1H), 2.40 (s, 2H), 2.10 – 1.92 (m, 2H), 1.56 (s, 2H), 1.25 (s, 6H), 0.76 (s, 9H); ^{13}C NMR (126 MHz, $CDCl_3$) δ 176.8, 167.6, 160.9, 149.3, 148.6, 136.8, 135.1, 130.9, 129.7, 126.1, 125.9, 124.0, 121.5, 119.1, 62.6, 55.8, 51.5, 51.4, 34.6, 32.2, 31.3, 31.1, 28.8, 28.7, 22.7; HRMS calcd for $C_{27}H_{37}N_4O_3$ $[M+H]^+$: 465.2860, found $[M+H]^+$: 465.2856.



C11:4-(4-((4-methoxy-4-oxobutyl)carbamoyl)-3-(2-methoxyethyl)-3,4-dihydroquinazolin-2-yl)butanoic acid:

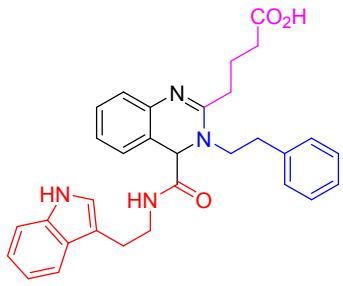
The product was synthesized according to procedure **B** in 1 mmol scale as light yellow solid (365 mg, 87% yield), M.P.= 77 – 78 °C. I.R. (thin film): 3217, 2932, 2351, 1731, 1556, 1167, 750 cm^{-1}. ^1H NMR (500 MHz,



CD_3OD) δ 7.91 (s, 1H), 7.46 – 7.39 (m, 2H), 7.31 (t, *J* = 7.6 Hz, 1H), 7.15 (d, *J* = 7.9 Hz, 1H), 5.55 (s, 1H), 4.18 – 4.09 (m, 1H), 3.71 – 3.66 (m, 3H), 3.62 (s, 3H), 3.33 (s, 3H), 3.28 – 3.13 (m, 2H), 3.05 – 2.96 (m, 1H), 2.83 – 2.73 (m, 1H), 2.45 – 2.34 (m, 2H), 2.25 (t, *J* = 7.4 Hz, 2H), 2.13 – 1.98 (m, 2H), 1.81 – 1.70 (m, 2H); ^{13}C NMR (126 MHz, CD_3OD) δ 180.3, 175.0, 169.7, 165.3, 132.2, 131.3, 128.5, 127.4, 119.7, 117.9, 79.5, 71.0, 64.5, 59.4, 52.6, 52.1, 40.0, 36.9, 31.7, 25.5, 24.6; HRMS calcd for $C_{21}H_{30}N_3O_6$ $[M+H]^+$: 420.2129, found $[M+H]^+$: 420.2128.

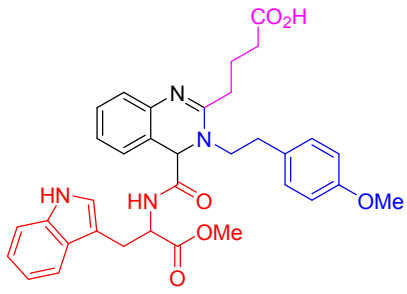
D8:4-(4-((2-(1H-indol-3-yl)ethyl)carbamoyl)-3-phenethyl-3,4-dihydroquinazolin-2-yl)butanoic acid:

The product was synthesized according to procedure **B** in 1 mmol scale as off white solid (240 mg, 47% yield), rotamers observed in 1:1 ratio M.P.= 161 – 163 °C. I.R. (thin film): 3213, 3054, 2923, 2358, 1650, 1556, 1454, 740 cm^{-1}. ^{1}H NMR (500 MHz, CD_3OD) (Major rotamer) δ 7.52 (d, *J* = 8.0 Hz, 1H), 7.37 (t, *J* = 7.4 Hz, 1H), 7.33 (d, *J* = 8.0 Hz, 1H), 7.23 – 7.18 (m, 5H), 7.15 – 7.06 (m, 6H), 6.99 (t, *J* = 7.4 Hz, 1H), 6.92 (s, 1H), 5.21 (s, 1H), 4.10 – 4.02 (m, 1H), 3.61 – 3.47 (m, 2H), 3.25 – 3.17 (m, 1H), 2.96 – 2.87 (m, 4H), 2.65 – 2.56 (m, 1H), 2.40 – 2.32 (m, 1H), 2.31 – 2.24 (m, 2H), 1.93 – 1.76 (m, 2H); ^{13}C NMR (126 MHz, CD_3OD) (Major rotamer) δ 180.2, 169.3, 164.7, 138.1, 138.1, 132.1, 131.3, 130.1, 129.9, 128.8, 128.5, 128.2, 127.4, 123.8, 122.4, 119.7, 119.2, 119.1, 117.9, 112.5, 112.4, 63.4, 53.3, 41.3, 36.9, 34.3, 31.5, 25.8, 24.3, 24.2; HRMS calcd for $C_{31}H_{32}N_4O_3$ $[M+H]^+$: 509.2547, found $[M+H]^+$: 509.2543.



F2:4-(4-((3-(1H-indol-3-yl)-1-methoxy-1-oxopropan-2-yl)carbamoyl)-3-(4-methoxyphenethyl)-3,4-dihydroquinazolin-2-yl)butanoic acid:

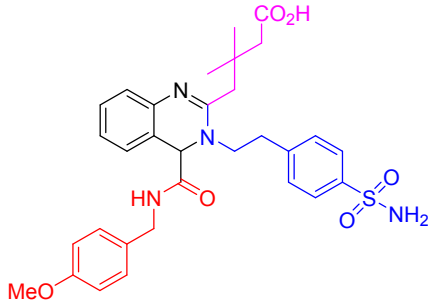
The product was synthesized according to procedure **B** in 1 mmol scale as light yellow solid (274 mg, 47% yield), rotamers observed in 1:1 ratio M.P.= 144 – 146 °C. I.R. (thin film): 3216, 2929, 2351, 1738, 1682, 1511, 1244, 1029, 750 cm^{-1}. ^{1}H NMR (500 MHz, DMSO-d_6) (Major rotamer) δ 10.93 (d, *J* = 2.0 Hz, 1H), 8.76 (d, *J* = 7.6 Hz, 1H), 7.50 (d, *J* = 8.0 Hz, 1H), 7.33 (s, 1H), 7.20 – 7.12 (m, 2H), 7.10 – 7.04 (m, 2H), 7.03-6.94 (m, 2H), 6.89-6.84 (m, 2H), 6.83-6.76 (m, 2H), 5.33 (s, 1H), 4.62-4.56 (m, 1H), 3.72 (s, 3H), 3.70 (s, 3H), 3.24 – 3.08 (m, 3H), 2.90-2.60



(m, 3H), 2.45-2.35 (m, 1H), 2.32-2.18 (m, 3H), 1.82-1.66 (m, 2H); ^{13}C NMR (126 MHz, DMSO-d_6) (Major rotamer) δ 174.7, 171.8, 169.8, 158.6, 157.9, 140.1, 136.1, 136.0, 129.9, 129.8, 128.6, 127.1, 126.9, 125.7, 124.0, 123.6, 121.0, 120.5, 118.5, 118.0, 113.9, 111.5, 109.2, 109.1, 60.3, 55.0, 52.8, 51.9, 50.4, 33.1, 32.2, 31.8, 27.1, 21.7; HRMS calcd for $C_{34}H_{37}N_4O_6$ $[M+H]^+$: 597.2708, found $[M+H]^+$: 597.2704.

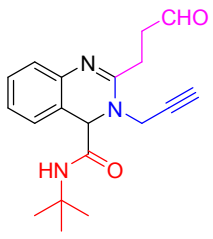
H13:4-(4-((4-methoxybenzyl)carbamoyl)-3-(4-sulfamoylphenethyl)-3,4-dihydroquinazolin-2-yl)-3,3-dimethylbutanoic acid:

The product was synthesized according to procedure **B** in 1 mmol scale as off white solid (285 mg, 48% yield), M.P.= more than 220 °C. I.R. (thin film): 3311, 3176, 2958, 2358, 1678, 1257, 1547, 1170, 750 cm^{-1}. ^{1}H NMR (500 MHz, DMSO-d_6) δ 9.08 (t, J = 5.6 Hz, 1H), 7.72 (d, J = 8.2 Hz, 2H), 7.43 (d, J = 8.2 Hz, 2H), 7.35 – 7.31 (m, 2H), 7.31 – 7.25 (m, 2H), 7.15 – 7.10 (m, 3H), 6.88 (d, J = 7.6 Hz, 1H), 6.84 (d, J = 8.6 Hz, 2H), 5.54 (s, 1H), 4.22 (t, J = 5.1 Hz, 2H), 4.20 – 4.10 (m, 1H), 3.71 (s, 3H), 3.26 – 3.14 (m, 2H), 3.07 – 2.92 (m, 2H), 2.71 (d, J = 14.1 Hz, 1H), 2.56 (d, J = 12.2 Hz, 1H), 2.32 (d, J = 14.1 Hz, 1H), 1.99 (d, J = 12.2 Hz, 1H), 1.15 (s, 3H), 1.05 (s, 3H); ^{13}C NMR (126 MHz, DMSO-d_6) δ 173.2, 168.7, 159.5, 158.3, 142.6, 141.7, 135.9, 130.6, 129.4, 128.5, 125.9, 125.7, 125.4, 119.5, 113.7, 60.7, 55.1, 50.7, 47.5, 41.8, 36.2, 32.8, 29.1, 28.0; HRMS calcd for $C_{31}H_{37}N_4O_6S$ $[M+H]^+$: 593.2428, found $[M+H]^+$: 593.2428.



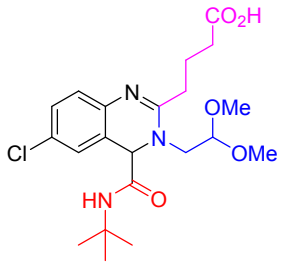
H17:3-(4-(tert-butylcarbamoyl)-3-(prop-2-yn-1-yl)-3,4-dihydroquinazolin-2-yl)propanoic acid:

The product was synthesized according to procedure **B** in 1 mmol scale as off white solid (246 mg, 72% yield), M.P.= 173 – 175 °C. I.R. (thin film): 3163, 2966, 2358, 1696, 1579, 750 cm^{-1}. ^{1}H NMR (500 MHz, CD_3OD) δ 7.90 (s, 1H), 7.42 – 7.36 (m, 2H), 7.34 – 7.29 (m, 1H), 7.16 (d, J = 7.9 Hz, 1H), 5.50 (s, 1H), 4.67 – 4.51 (m, 2H), 3.16 – 3.11 (m, 2H), 2.84 – 2.71 (m, 2H), 1.31 (s, 9H); ^{13}C NMR (126 MHz, CD_3OD) δ 177.5, 168.6, 165.2, 131.9, 131.2, 128.7, 127.0, 120.2, 118.4, 79.5, 78.3, 76.3, 64.8, 52.8, 42.4, 34.0, 28.6; HRMS calcd for $C_{19}H_{24}N_3O_3$ $[M+H]^+$: 342.1812, found $[M+H]^+$: 342.1810.



H21:N-(tert-butyl)-10-chloro-7-(2,2-dimethoxyethyl)-2,6-dioxo-1,2,3,4,5,6,7,8-octahydrobenzo[b][1,5]diazecine-8-carboxamide:

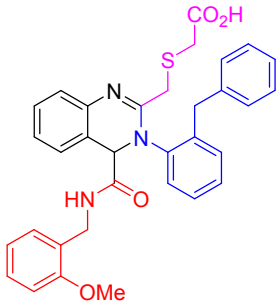
The product was synthesized according to procedure **B** in 1 mmol scale as off white solid (180 mg,



41% yield), M.P.= 118 – 120 °C. I.R. (thin film): 3217, 2934, 2354, 1681, 1556, 1066, 827 cm^{-1}. ^1H NMR (500 MHz, CD_3OD) δ 7.47 (s, 1H), 7.40 (d, J = 8.3 Hz, 1H), 7.15 (d, J = 8.4 Hz, 1H), 5.59 – 5.52 (m, 1H), 4.65 (s, 1H), 4.13 (dd, J = 15.1, 3.6 Hz, 1H), 3.48 (s, 3H), 3.47 (s, 3H), 3.42 (dd, J = 15.2, 3.5 Hz, 1H), 3.10 – 2.66 (m, 2H), 2.57 – 2.35 (m, 2H), 2.16 – 1.97 (m, 2H), 1.31 (s, 9H); ^{13}C NMR (126 MHz, CD_3OD) δ 179.8, 168.3, 165.7, 133.4, 131.4, 131.2, 121.9, 119.9, 104.1, 66.9, 65.2, 56.6, 56.4, 54.0, 52.9, 28.6, 23.9; HRMS calcd for $C_{21}H_{31}ClN_3O_5$ $[M+H]^+$: 440.1947, found $[M+H]^+$: 440.1946.

J12:2-(((3-(2-benzylphenyl)-4-((2-methoxybenzyl)carbamoyl)-3,4-dihydroquinazolin-2-yl)methyl)thio)acetic acid:

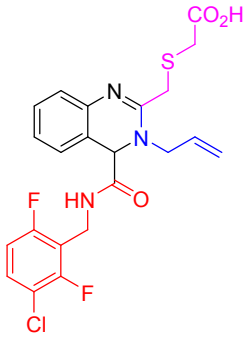
The product was synthesized according to procedure **B** in 1 mmol scale as off white solid (475 mg,



84% yield), M.P. = 173 – 174 °C. I.R. (thin film): 3232, 2915, 2358, 1681 1564, 1241, 750 cm^{-1}. ^1H NMR (500 MHz, CD_3OD) δ 7.63 (d, *J* = 7.4 Hz, 1H), 7.56 (td, *J* = 7.4, 1.8 Hz, 1H), 7.52 – 7.48 (m, 1H), 7.48 – 7.43 (m, 1H), 7.38 – 7.30 (m, 3H), 7.25 (td, *J* = 8.1, 1.7 Hz, 1H), 7.19 (d, *J* = 7.6 Hz, 1H), 7.14 – 7.01 (m, 5H), 6.92 (d, *J* = 8.2 Hz, 1H), 6.85 – 6.79 (m, 3H), 5.28 (s, 1H), 4.33 – 4.25 (m, 2H), 4.00 (d, *J* = 15.2 Hz, 1H), 3.90 (d, *J* = 15.2 Hz, 1H), 3.73 (s, 3H), 3.47 (d, *J* = 15.7 Hz, 1H), 3.20 (d, *J* = 15.7 Hz, 1H); ^{13}C NMR (126 MHz, CD_3OD) δ 176.0, 169.4, 161.8, 158.8, 139.6, 139.0, 138.6, 134.2, 132.2, 131.5, 130.4, 130.3, 130.2, 130.1, 130.0, 129.5, 129.1, 128.2, 127.3, 126.4, 121.2, 111.4, 66.7, 55.8, 40.2, 39.7, 37.1; HRMS calcd for $C_{33}H_{32}N_3O_4S$ $[M+H]^+$: 566.2108, found $[M+H]^+$: 566.2105.

J23:2-(((3-allyl-4-((3-chloro-2,6-difluorobenzyl)carbamoyl)-3,4-dihydroquinazolin-2-yl)methyl)thio)acetic acid:

The product was synthesized according to procedure **B** in 1 mmol scale as off white solid (269 mg,

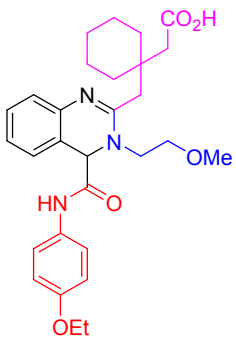


56% yield), M.P.= 180 – 182 °C. I.R. (thin film): 3176, 3053, 2944, 2823, 2351, 1683, 1579, 684 cm^{-1}. ^1H NMR (500 MHz, DMSO-d_6) δ 8.87 (t, *J* = 5.2 Hz, 1H), 7.62 – 7.56 (m, 1H), 7.21 – 7.14 (m, 3H), 7.02 (t, *J* = 7.4 Hz,

1H), 6.96 (d, J = 8.1 Hz, 1H), 5.86 – 5.75 (m, 1H), 5.19 (d, J = 17.2 Hz, 1H), 5.14 (d, J = 10.1 Hz, 1H), 5.09 (s, 1H), 4.38 – 4.20 (m, 3H), 3.80 – 3.73 (m, 1H), 3.73 – 3.64 (m, 2H), 3.46 (d, J = 15.5 Hz, 1H), 3.37 (d, J = 15.4 Hz, 1H); ^{13}C NMR (126 MHz, DMSO-d_6) δ 171.1, 169.1, 169.0, 159.4 (dd, J = 248.7, 6.8 Hz), 156.1 (dd, J = 250.1, 8.7 Hz), 156.0, 138.8, 132.6, 129.9 (m), 128.8, 125.6, 124.9, 120.6, 118.6, 115.7 (dd, J = 19.9, 3.0 Hz), 115.4 (dd, J = 15.6, 3.8 Hz), 112.5 (d, J = 27.7 Hz), 60.7, 52.4, 33.6, 21.3; HRMS calcd for $C_{22}H_{21}ClF_2N_3O_3S$ $[M+H]^+$: 480.0955, found $[M+H]^+$: 480.0954.

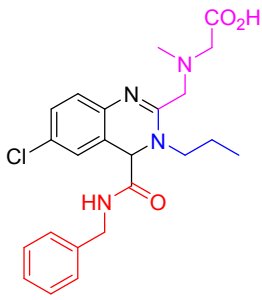
K7:2-(1-((4-((4-ethoxyphenyl)carbamoyl)-3-(2-methoxyethyl)-3,4-dihydroquinazolin-2-yl)methyl)cyclohexyl)acetic acid:

The product was synthesized according to procedure **B** in 1 mmol scale as off white solid (360 mg, 71% yield), M.P.= 141 – 144 °C. I.R. (thin film): 3163, 2966, 2358, 1696, 1579, 750 cm^{-1}. ^{1}H NMR (500 MHz, $CDCl_3$) δ 11.59 (s, 1H), 7.76 (d, J = 7.6 Hz, 1H), 7.64 (d, J = 8.9 Hz, 2H), 7.12 (dt, J = 34.2, 7.4 Hz, 2H), 6.99 (d, J = 7.4 Hz, 1H), 6.77 (d, J = 8.9 Hz, 2H), 6.51 (s, 1H), 4.20 (t, J = 10.6 Hz, 1H), 3.94 (q, J = 7.0 Hz, 2H), 3.81 (t, J = 9.1 Hz, 1H), 3.61 – 3.47 (m, 2H), 3.24 (s, 3H), 2.92 (dd, J = 53.2, 14.1 Hz, 2H), 2.72 (dd, J = 42.8, 12.6 Hz, 2H), 2.00 (d, J = 10.5 Hz, 1H), 1.78 (d, J = 9.4 Hz, 2H), 1.69 – 1.58 (m, 2H), 1.57 – 1.48 (m, 3H), 1.48 – 1.40 (m, 1H), 1.35 (t, J = 7.0 Hz, 4H), 1.30 – 1.21 (m, 1H); ^{13}C NMR (126 MHz, $CDCl_3$) δ 178.5, 166.4, 161.3, 155.6, 131.6, 129.4, 126.9, 121.8, 118.8, 117.3, 114.5, 68.9, 63.6, 62.3, 59.2, 51.0, 40.0, 37.7, 37.0, 25.8, 21.7, 14.8; HRMS calcd for $C_{29}H_{38}N_3O_5$ $[M+H]^+$: 508.2806, found $[M+H]^+$: 508.2801.



N7:N-((4-(benzylcarbamoyl)-6-chloro-3-propyl-3,4-dihydroquinazolin-2-yl)methyl)-N-methylglycine:

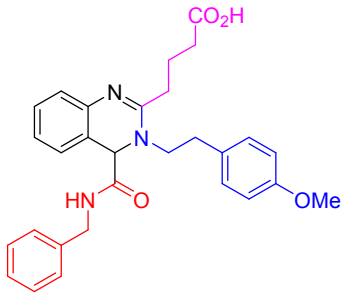
The product was synthesized according to procedure **B** in 1 mmol scale as white yellow solid (310 mg, 70% yield), M.P.= 134 – 137 °C. I.R. (thin



film): 3201, 2930, 2359, 1681, 1551, 697 cm^{-1}. ^1H NMR (500 MHz, CD_3OD) δ 7.91 (s, 1H), 7.46 – 7.32 (m, 3H), 7.31 – 7.21 (m, 4H), 7.18 (d, *J* = 7.1 Hz, 2H), 5.44 (s, 1H), 4.43 – 4.32 (m, 2H), 4.08 – 3.94 (m, 2H), 3.65 – 3.53 (m, 1H), 3.39 – 3.33 (m, 2H), 2.57 (s, 3H), 1.77 – 1.59 (m, 2H), 0.94 (t, *J* = 7.3 Hz, 3H); ^{13}C NMR (126 MHz, CD_3OD) δ 178.1, 169.5, 162.8, 139.2, 133.3, 131.2, 129.7, 128.6, 128.6, 127.1, 121.7, 121.0, 79.5, 63.8, 62.4, 53.1, 44.5, 43.2, 21.7, 11.1; HRMS calcd for $C_{23}H_{28}ClN_4O_3$ $[M+H]^+$: 443.1844, found $[M+H]^+$: 443.1844.

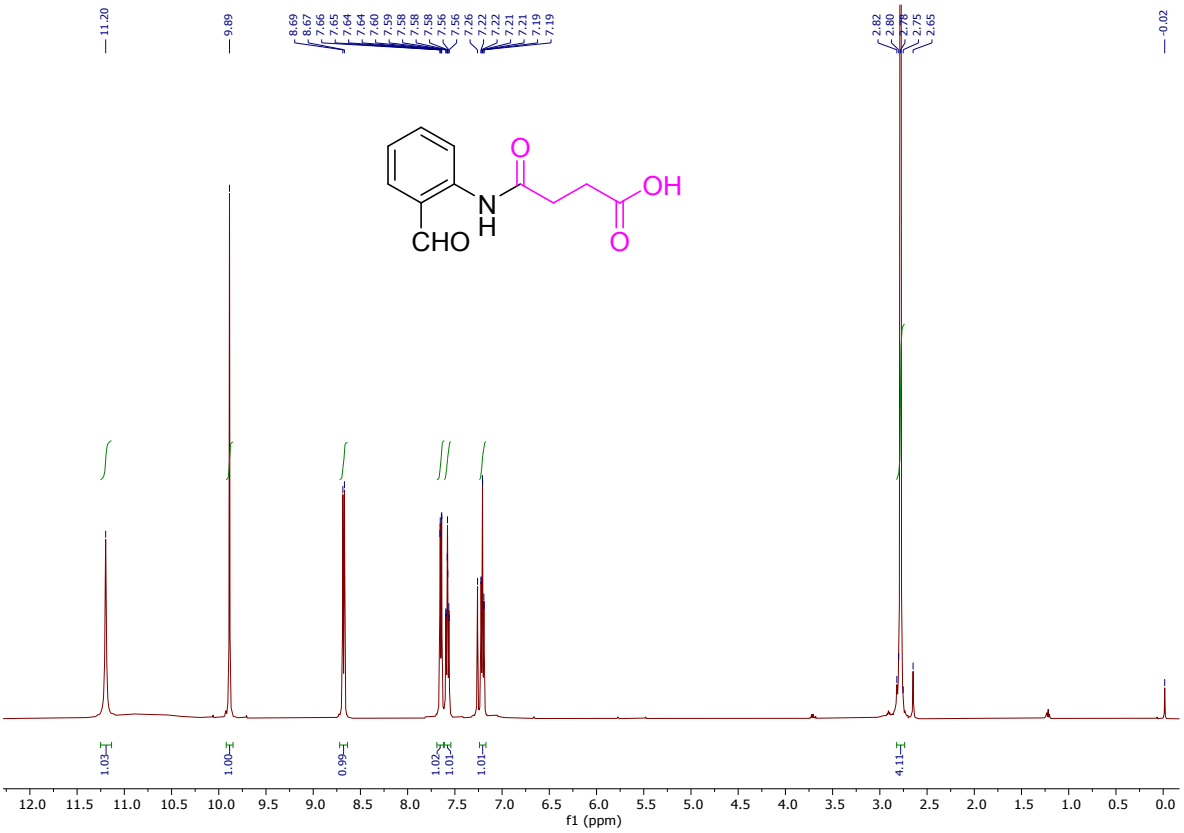
P2:4-(4-(benzylcarbamoyl)-3-(4-methoxyphenethyl)-3,4-dihydroquinazolin-2-yl)butanoic acid:

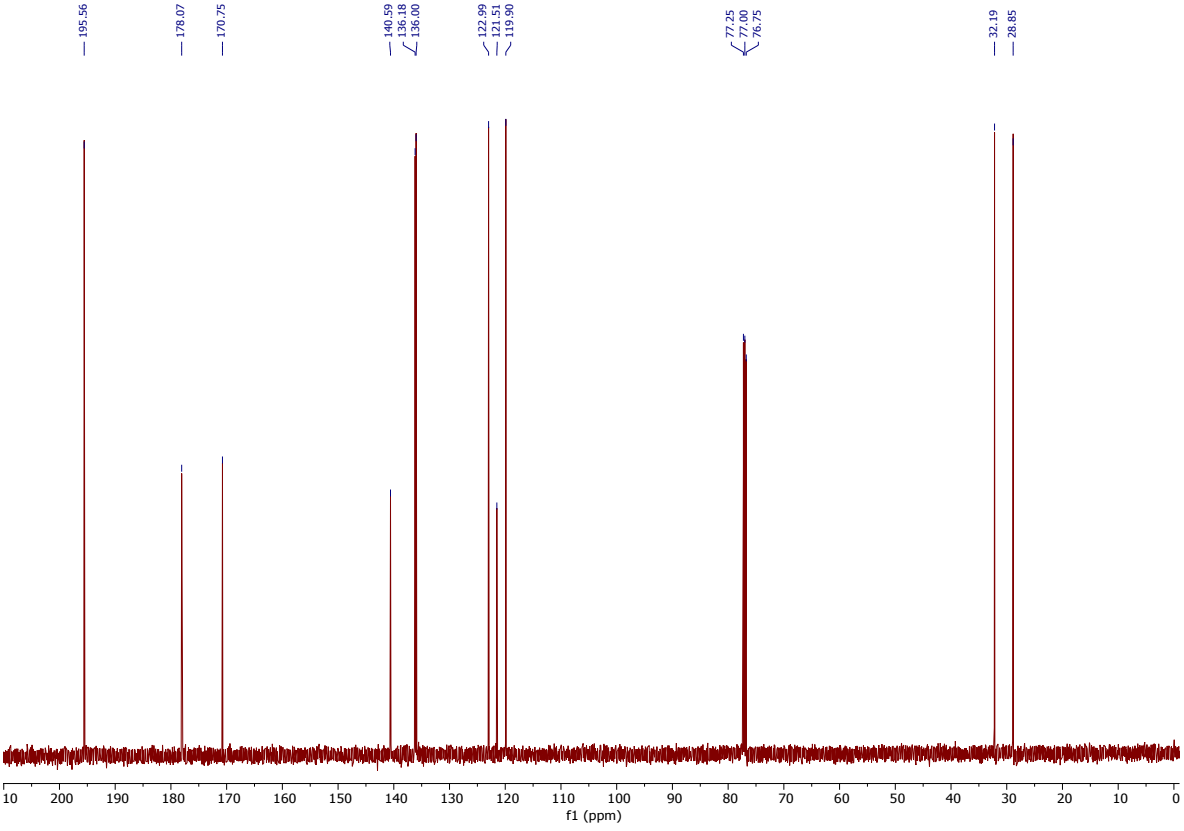
The product was synthesized according to procedure **B** in 1 mmol scale as off white solid (407 mg, 84% yield), M.P.= 86 – 88 °C. I.R. (thin film): 3212, 2930, 2358, 1681, 1511, 1245, 697 cm^{-1}. ^1H NMR (500 MHz, CD_3OD) δ 7.47 – 7.40 (m, 1H), 7.38 – 7.28 (m, 2H), 7.29 – 7.20 (m, 3H), 7.19 – 7.11 (m, 3H), 7.11 – 7.05 (m, 2H), 6.82 – 6.75 (m, 2H), 5.42 (s, 1H), 4.36 (s, 2H), 4.25 – 4.16 (m, 1H), 3.72 (s, 3H), 3.58 – 3.49 (m, 1H), 3.02 – 2.90 (m, 2H), 2.75 – 2.65 (m, 1H), 2.51 – 2.42 (m, 1H), 2.37 – 2.24 (m, 2H), 2.03 – 1.83 (m, 2H); ^{13}C NMR (126 MHz, CD_3OD) δ 180.2, 169.3, 164.9, 160.4, 139.2, 132.3, 131.5, 131.2, 129.8, 129.7, 128.6, 128.5, 127.5, 119.2, 118.1, 115.4, 115.2, 63.6, 55.7, 55.6, 54.0, 44.5, 36.8, 33.6, 24.3; HRMS calcd for $C_{29}H_{32}N_3O_4$ $[M+H]^+$: 486.2387, found $[M+H]^+$: 486.2383.



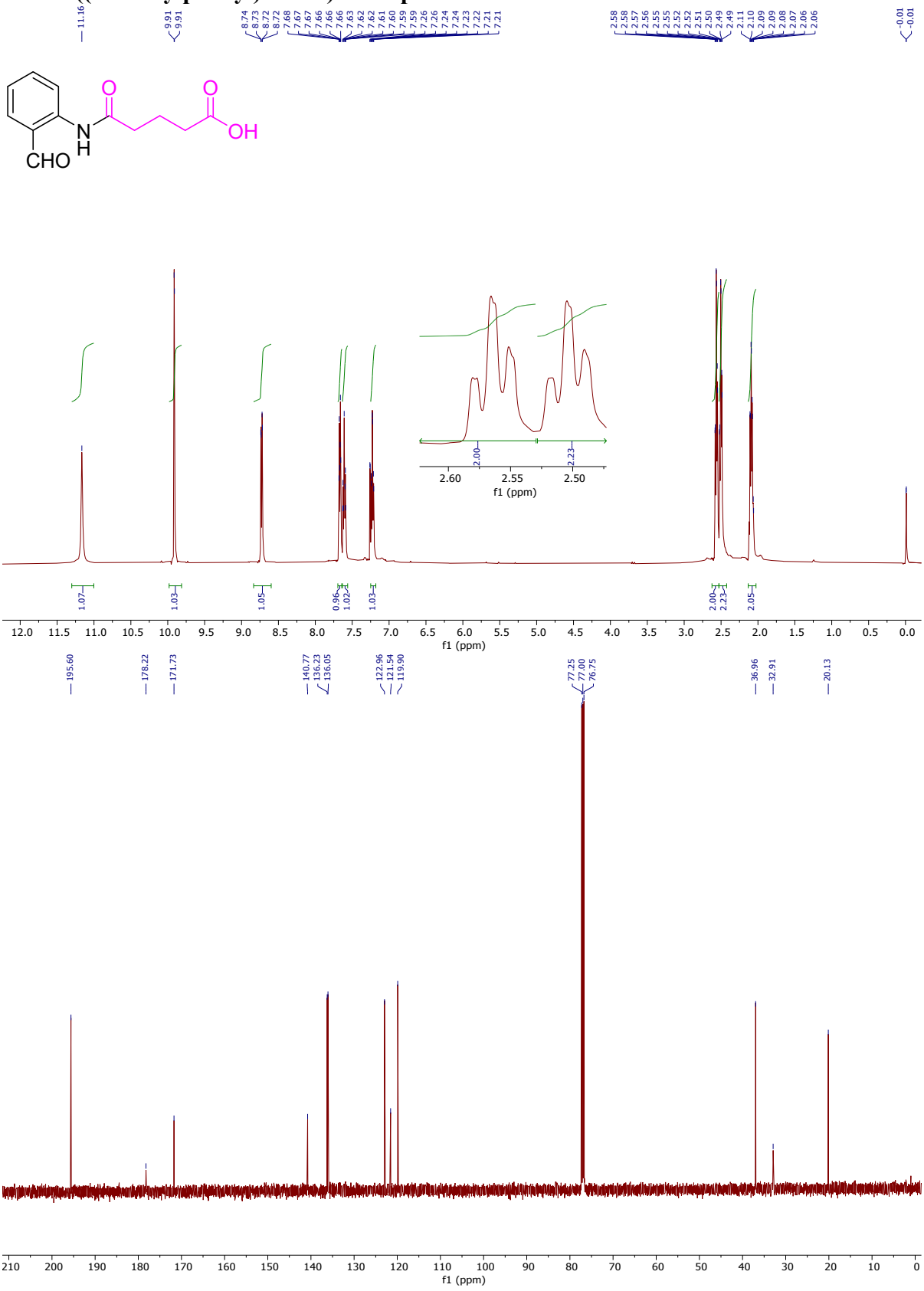
^{1}H, ^{13}C NMR and IR spectra of mg scale reaction

C-1: 4-((2-formylphenyl)amino)-4-oxobutanoic acid

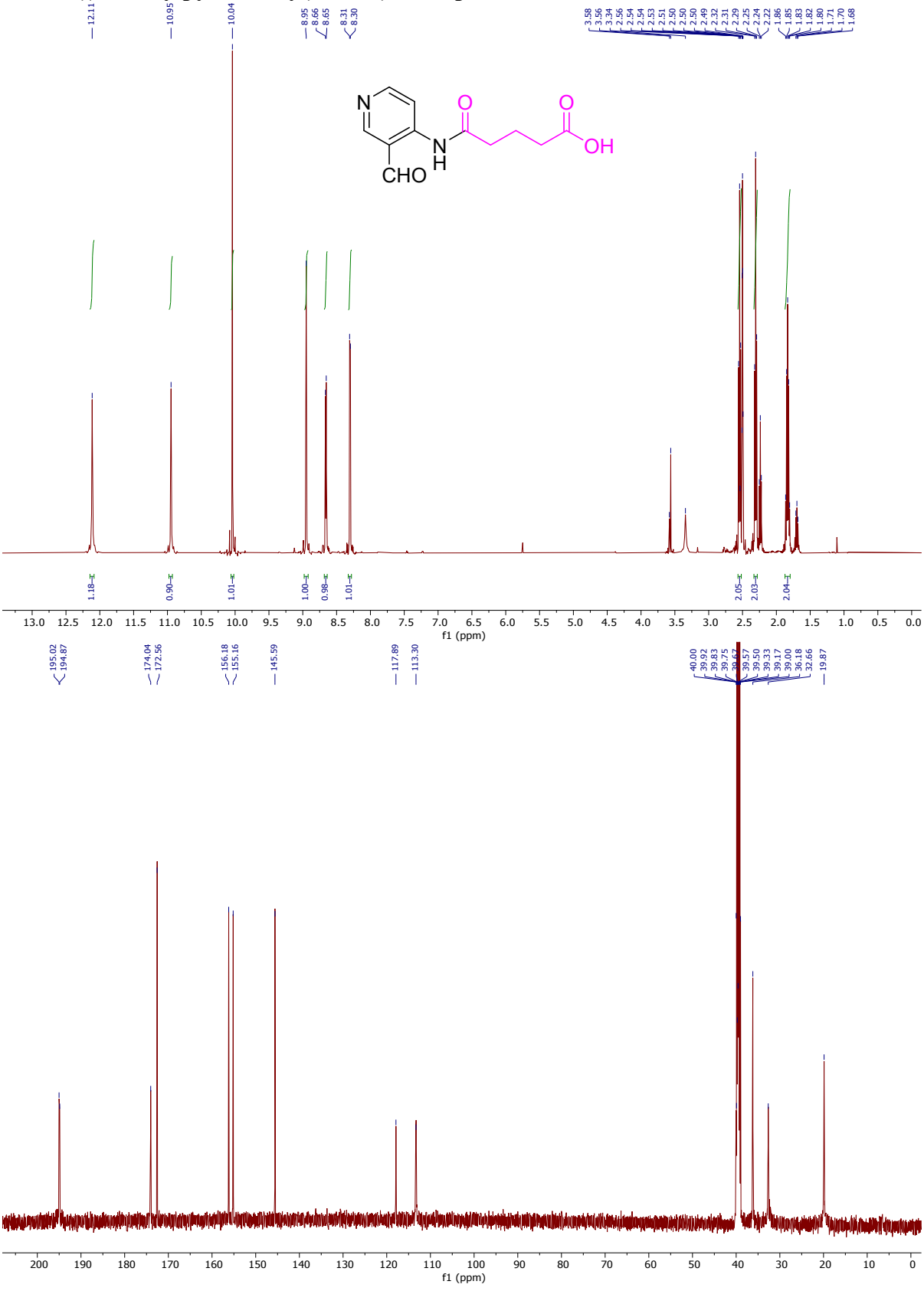




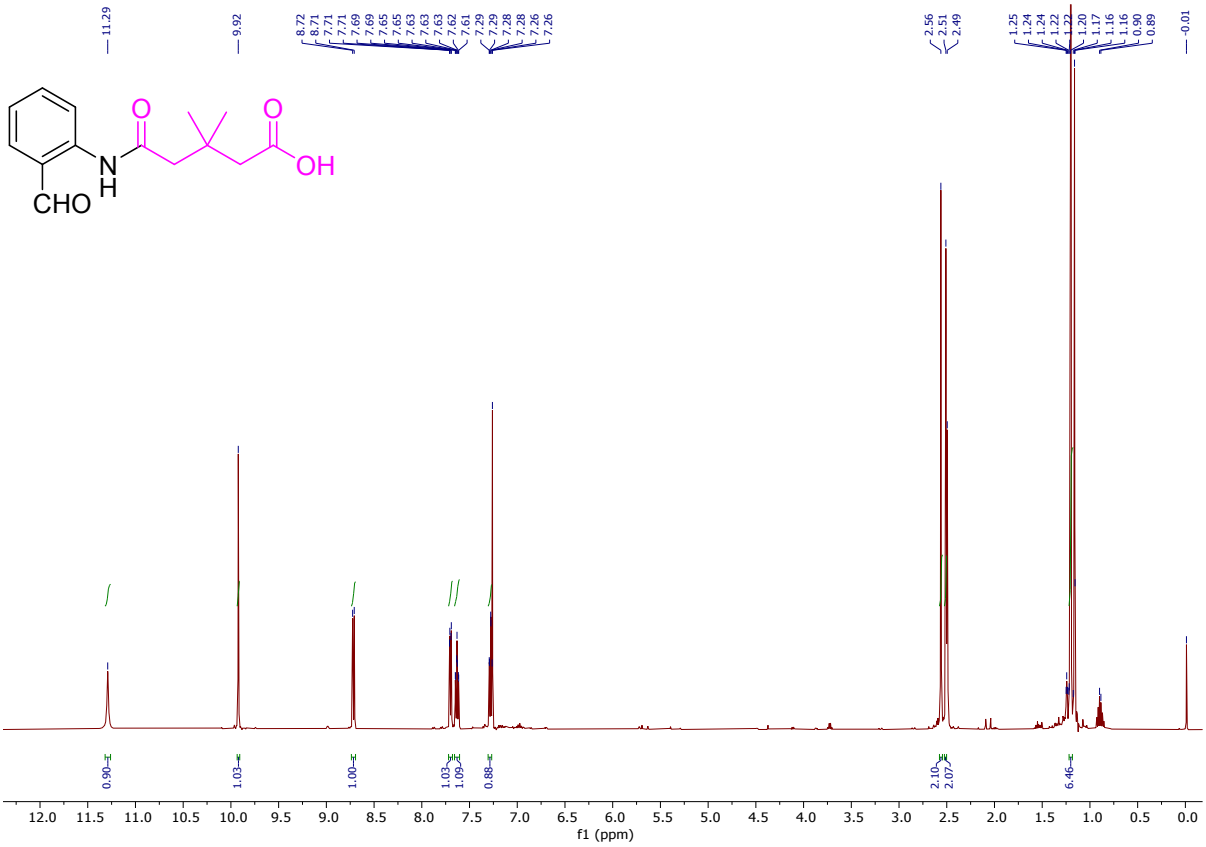
C-2: 5-((2-formylphenyl)amino)-5-oxopentanoic acid

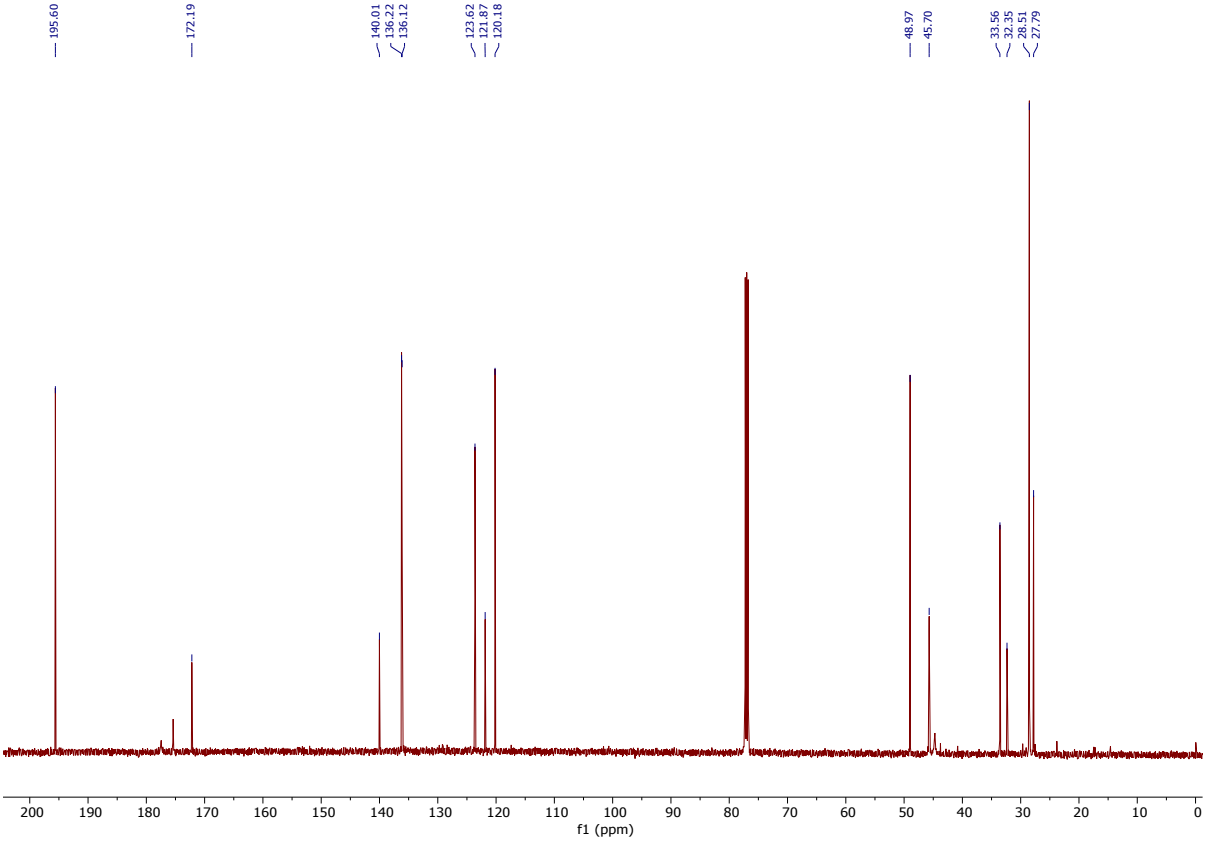


C-3: 5-((3-formylpyridin-4-yl)amino)-5-oxopentanoic acid

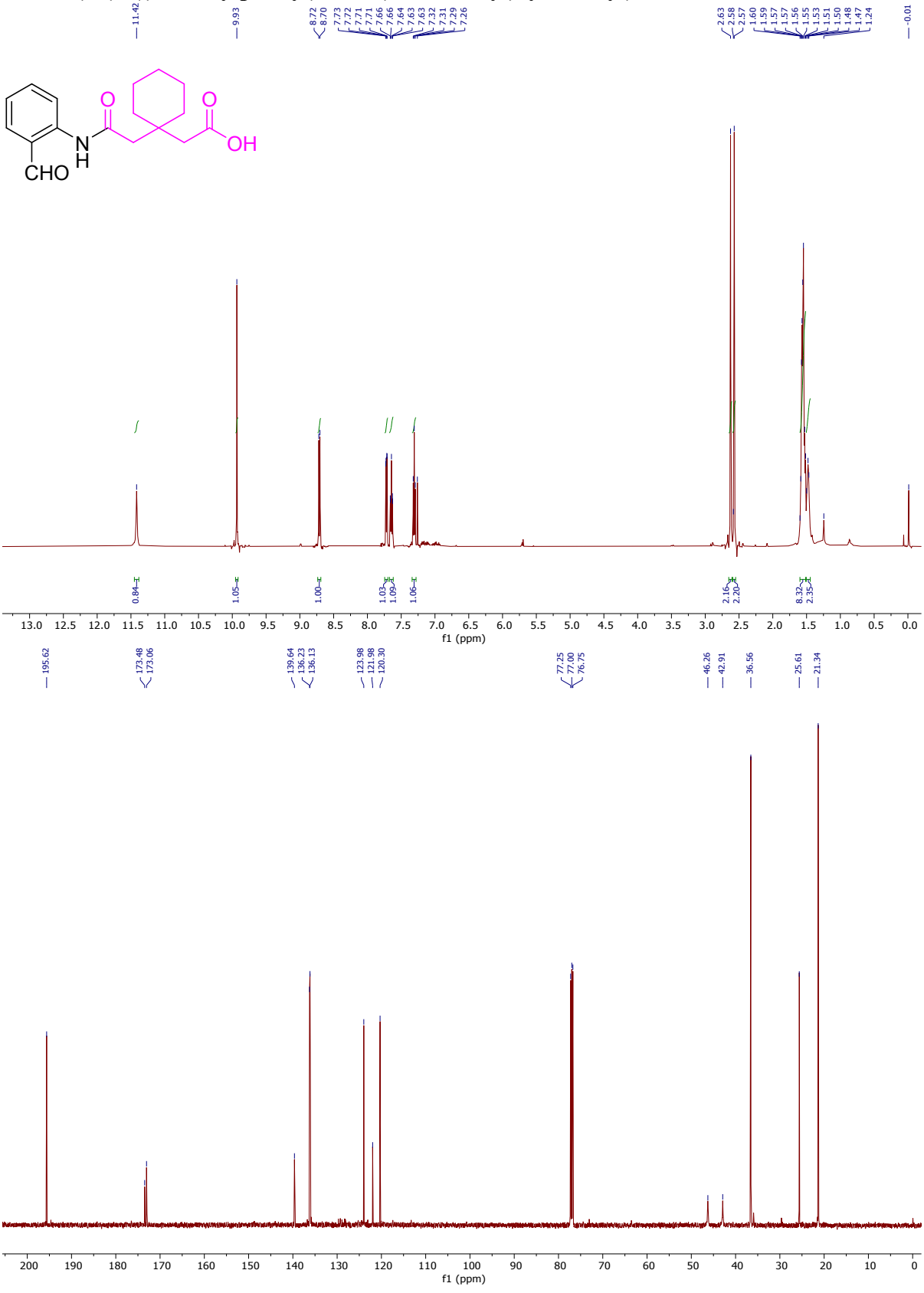


C-4: 5-((2-formylphenyl)amino)-3,3-dimethyl-5-oxopentanoic acid

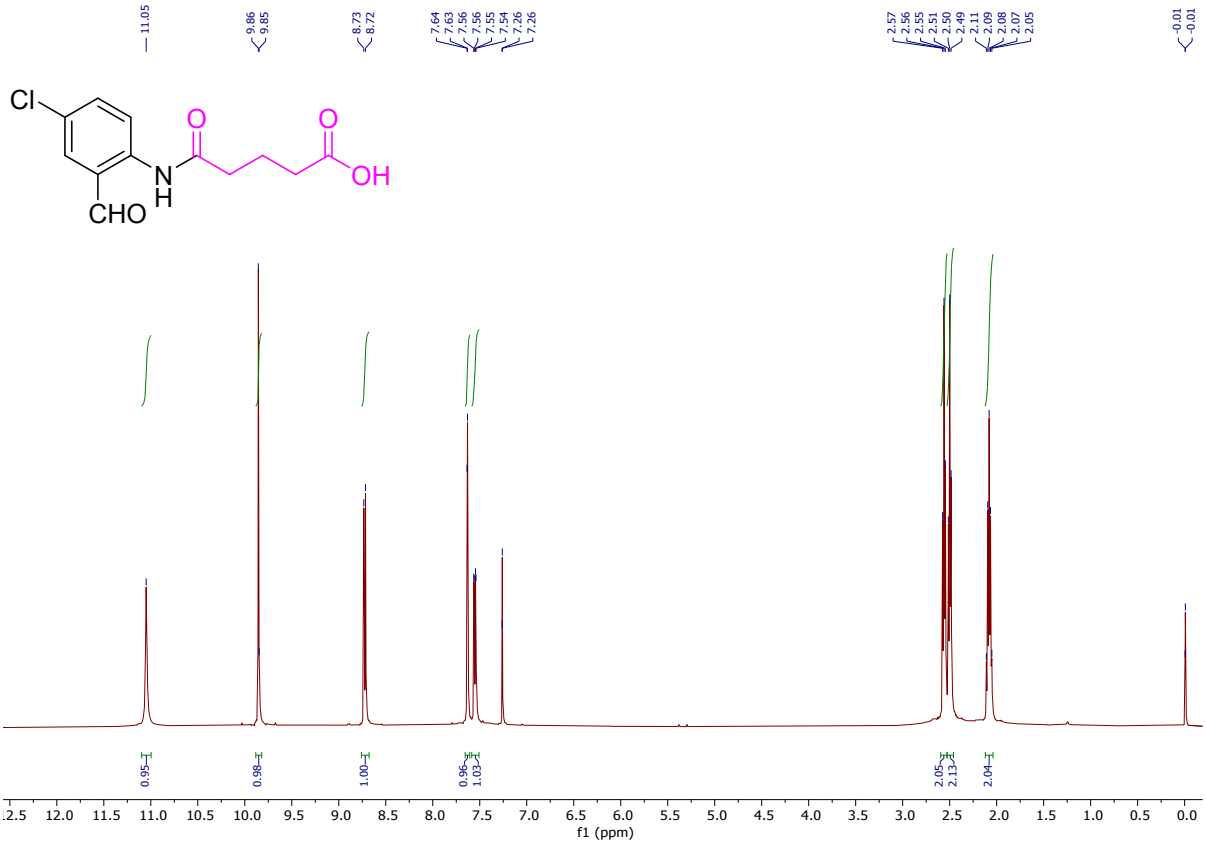


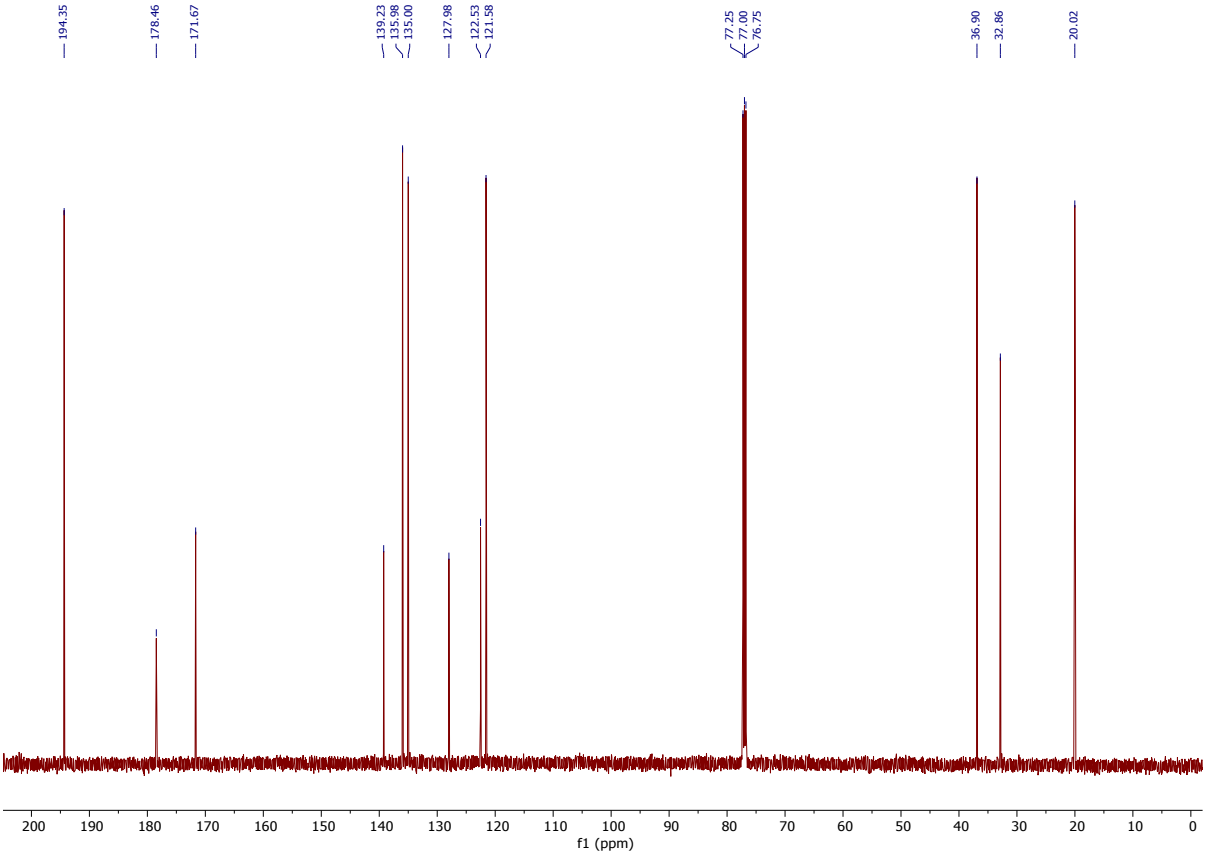


C-5: 2-(1-(2-((2-formylphenyl)amino)-2-oxoethyl)cyclohexyl)acetic acid

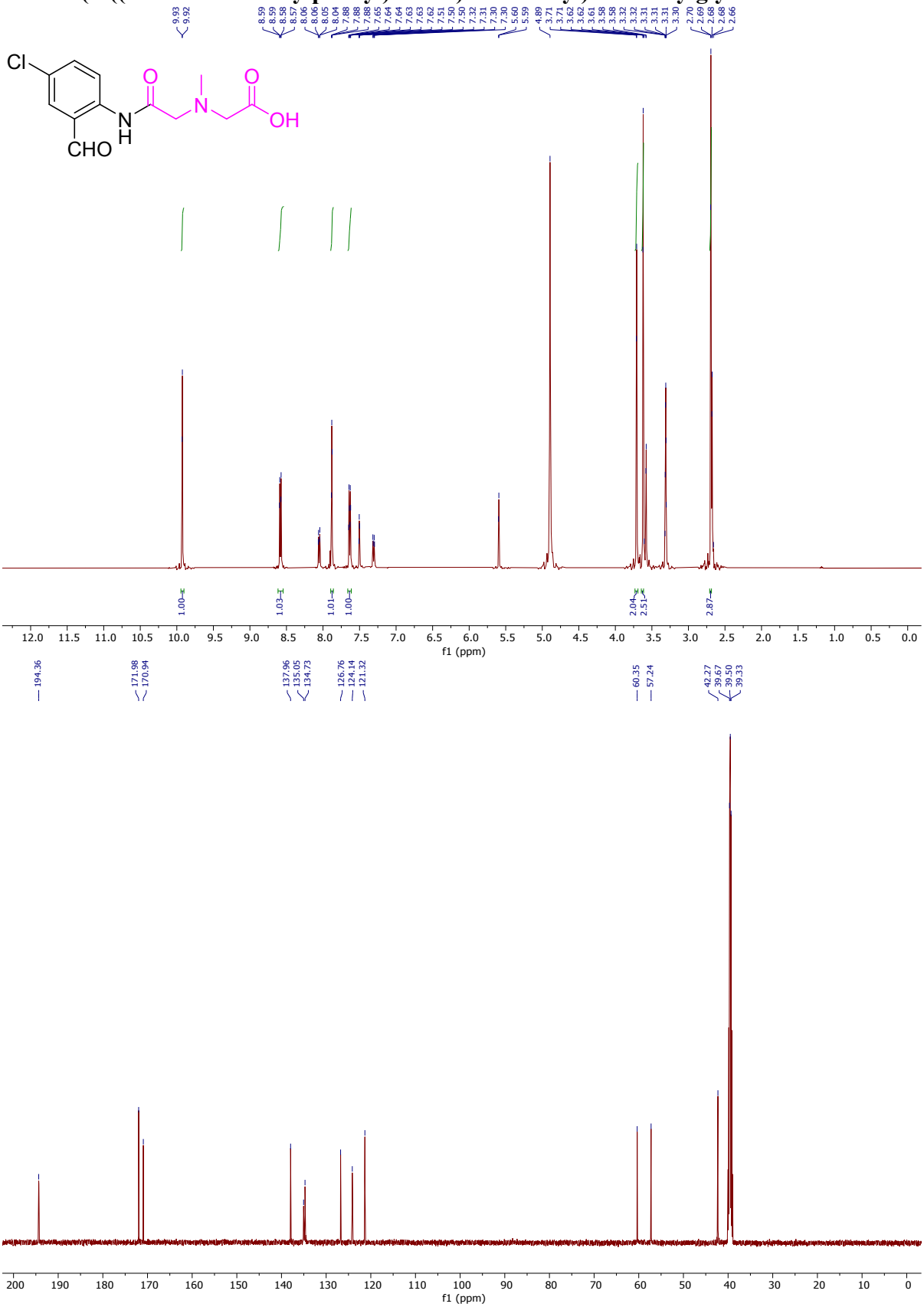


C-6: 5-((4-chloro-2-formylphenyl)amino)-5-oxopentanoic acid

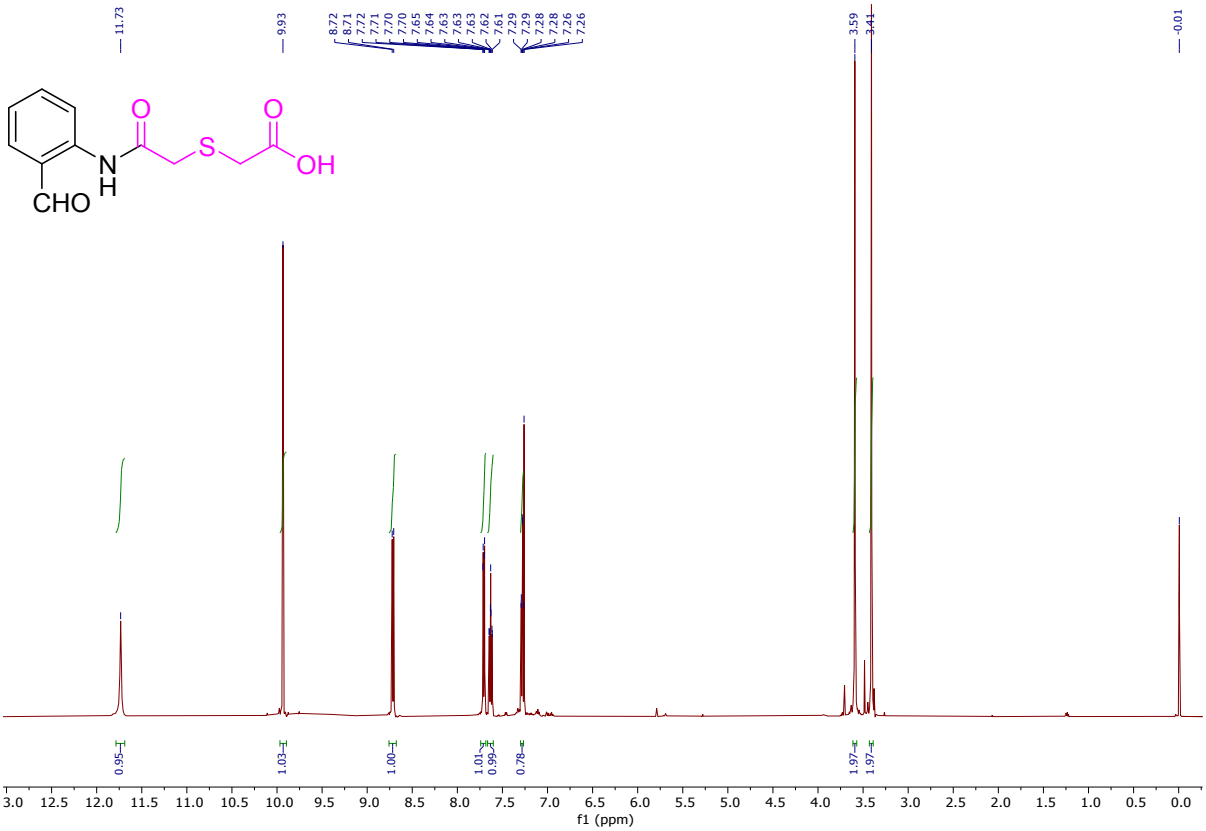


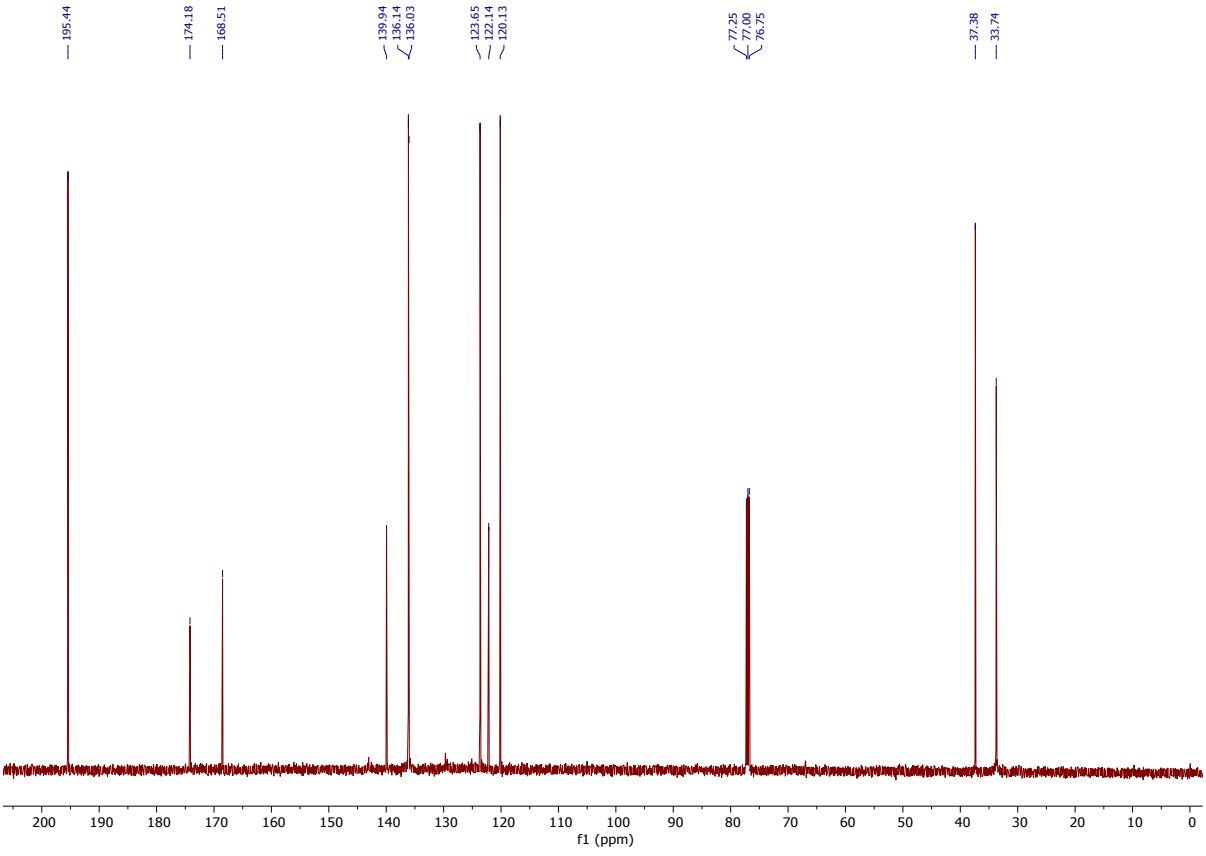


C-7: N-(2-((4-chloro-2-formylphenyl)amino)-2-oxoethyl)-N-methylglycine

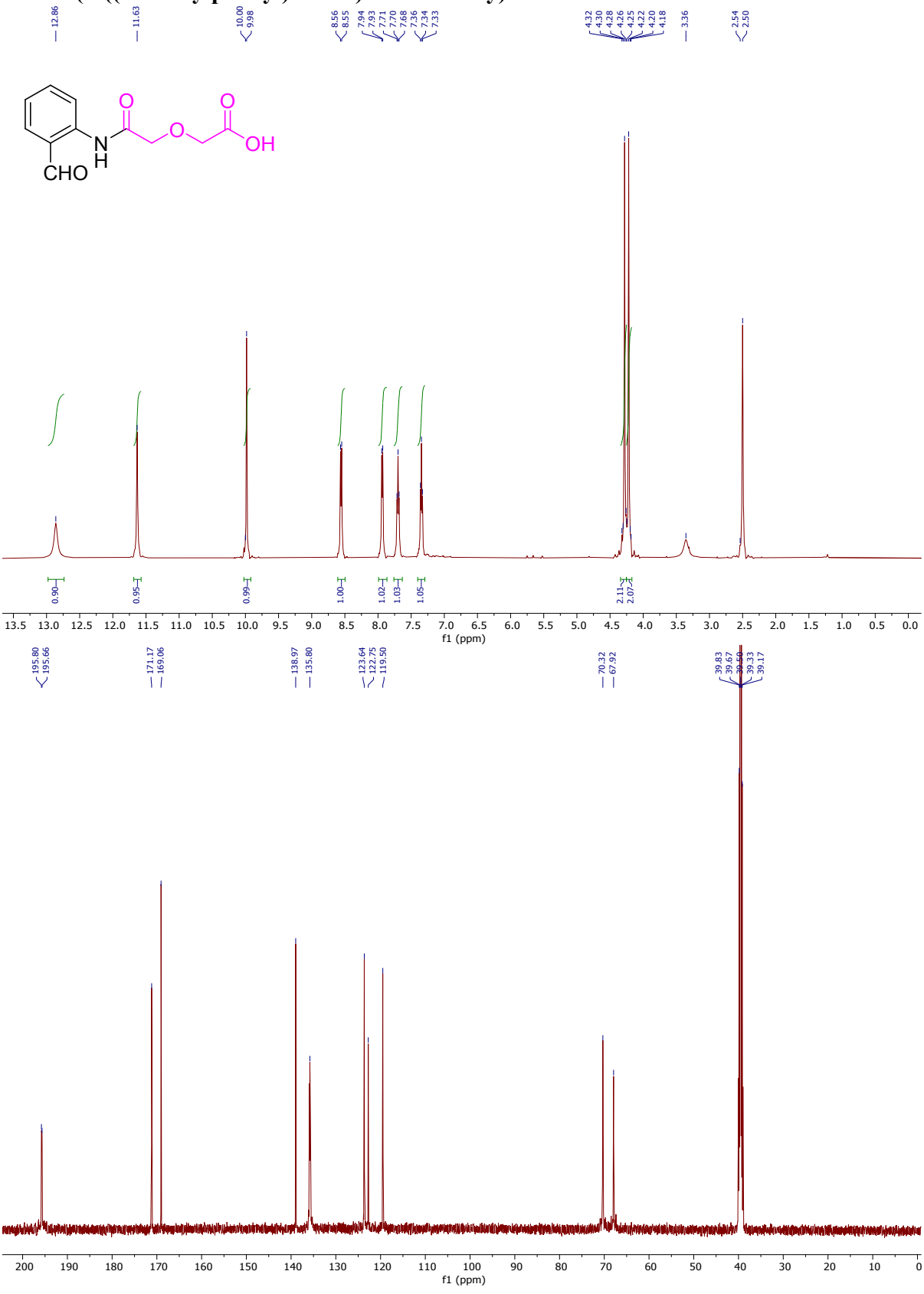


C-8: 2-((2-((2-formylphenyl)amino)-2-oxoethyl)thio)acetic acid

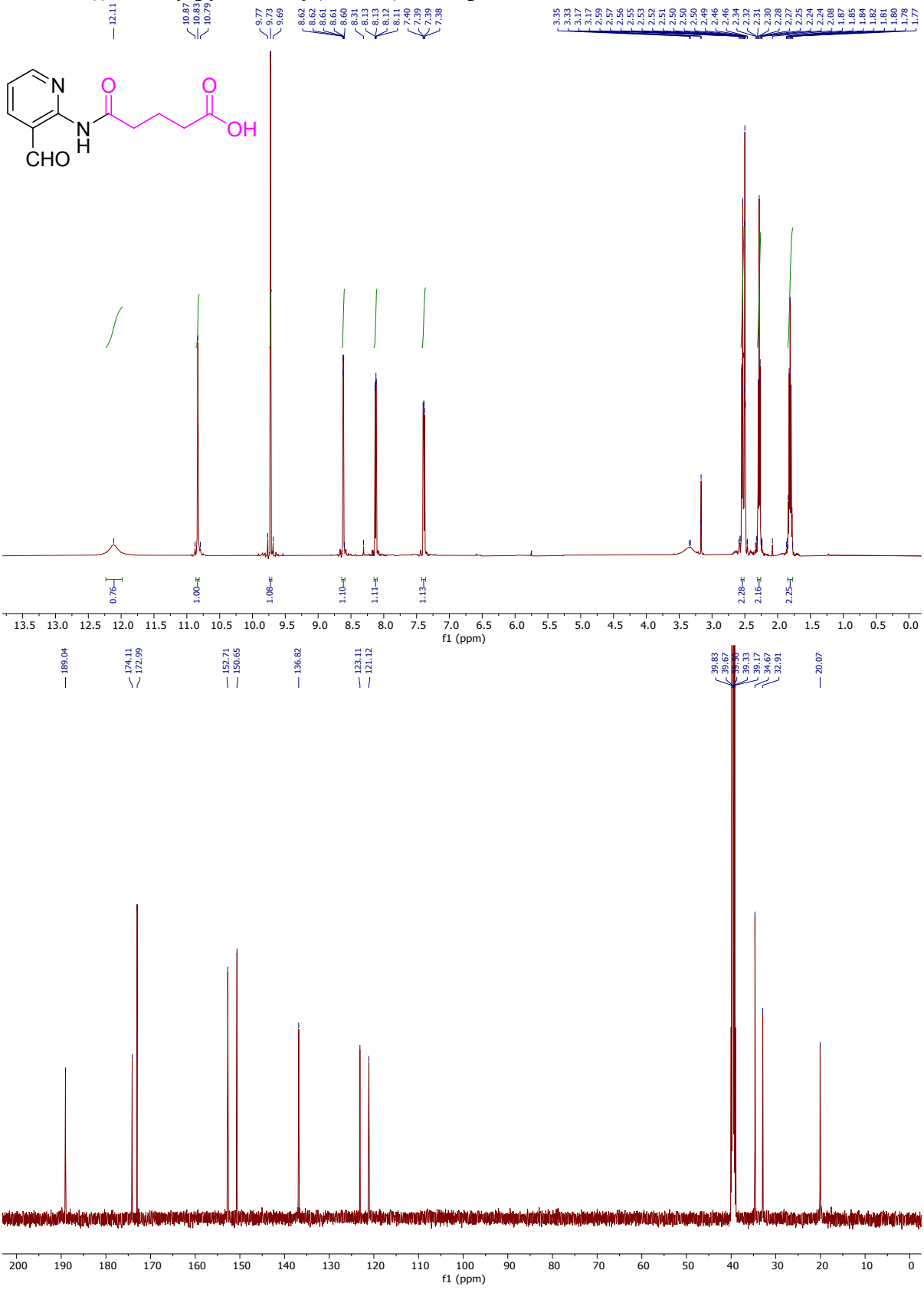




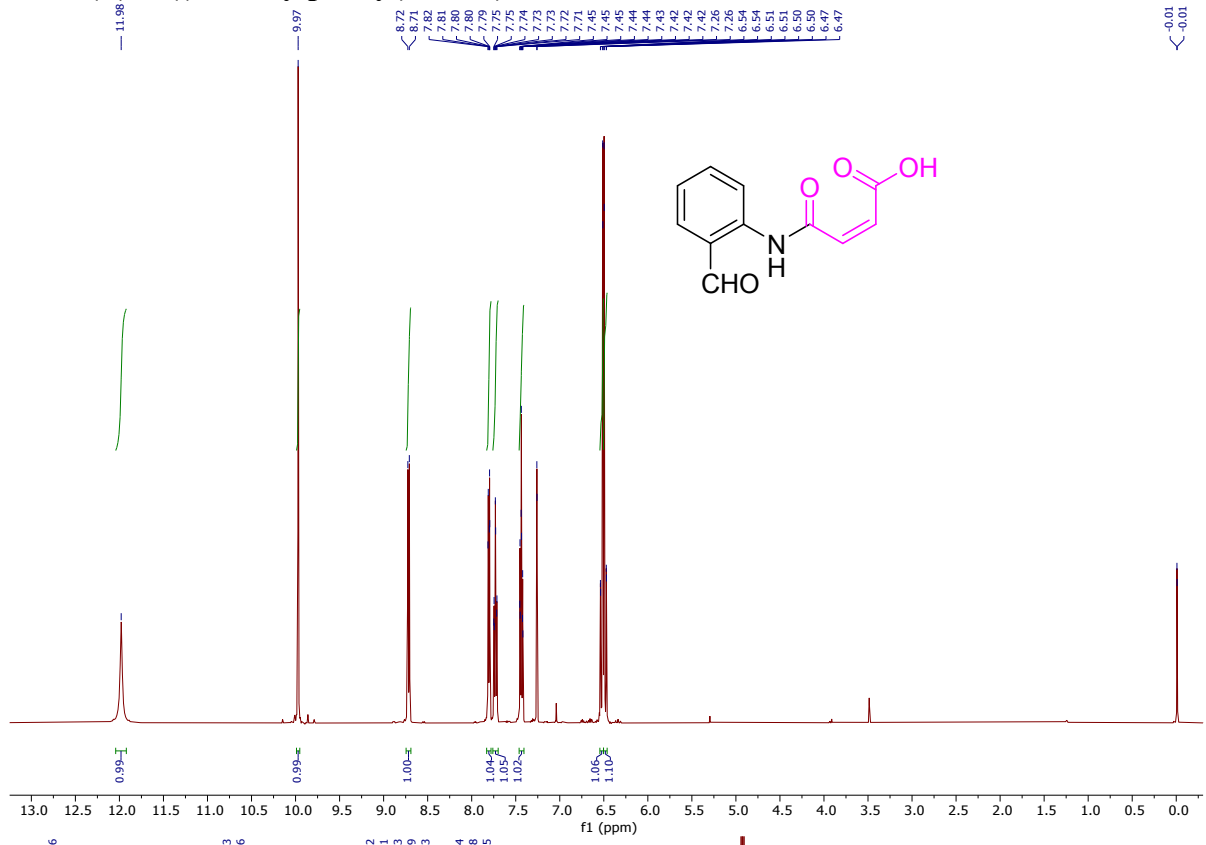
C-9: 2-(2-((2-formylphenyl)amino)-2-oxoethoxy)acetic acid

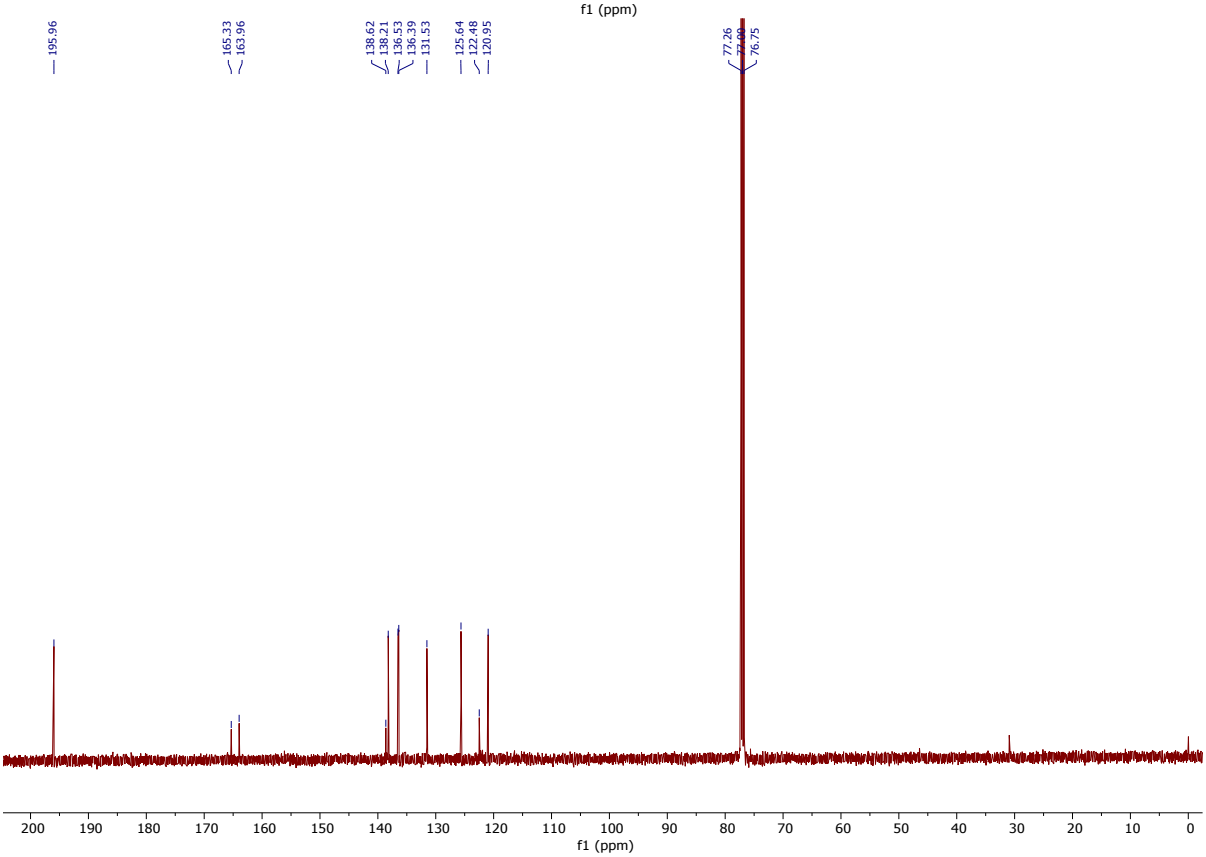


C-10: 5-((3-formylpyridin-2-yl)amino)-5-oxopentanoic acid

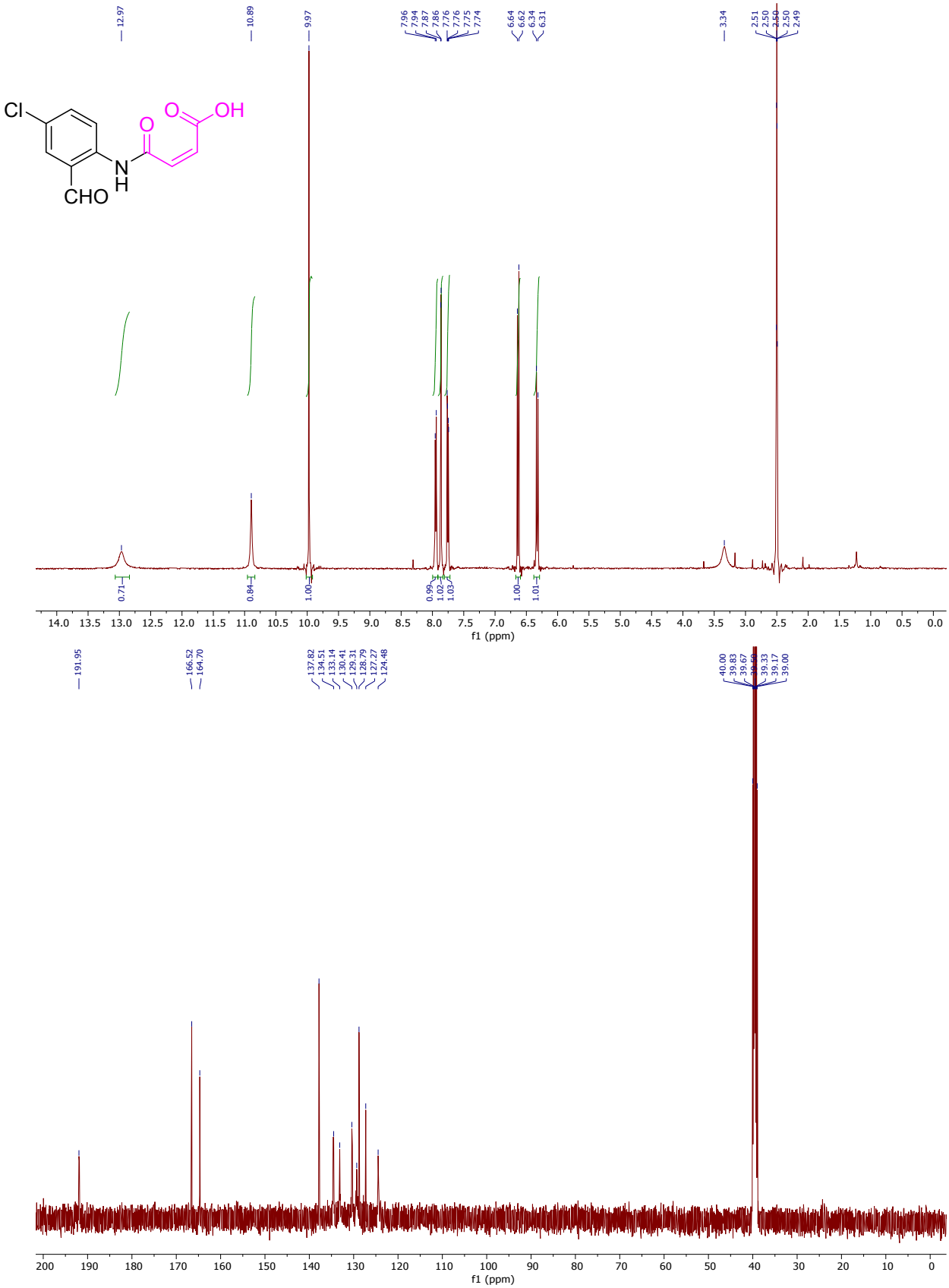


C-11: (Z)-4-((2-formylphenyl)amino)-4-oxobut-2-enoic acid

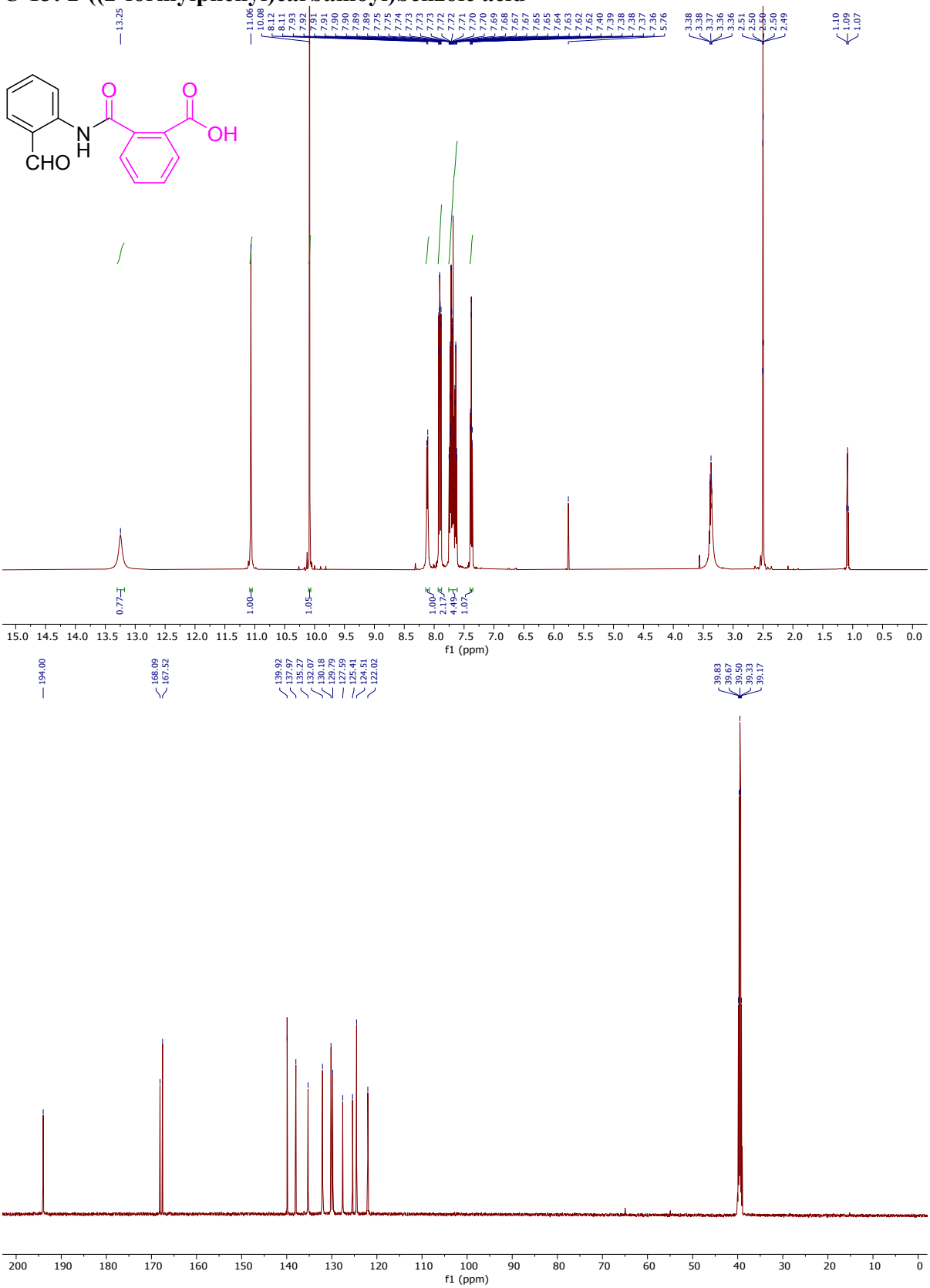




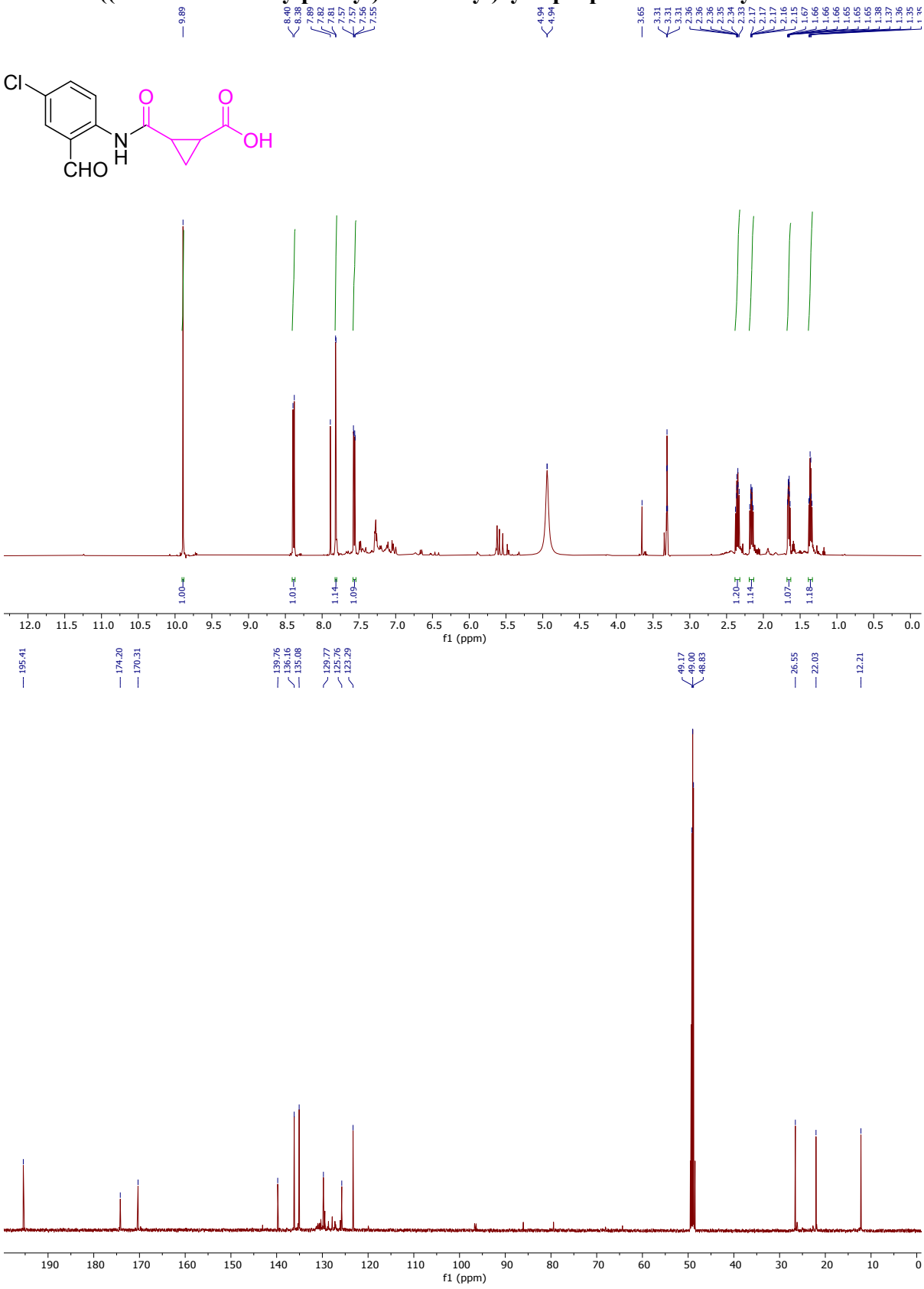
C-12: (Z)-4-((4-chloro-2-formylphenyl)amino)-4-oxobut-2-enoic acid



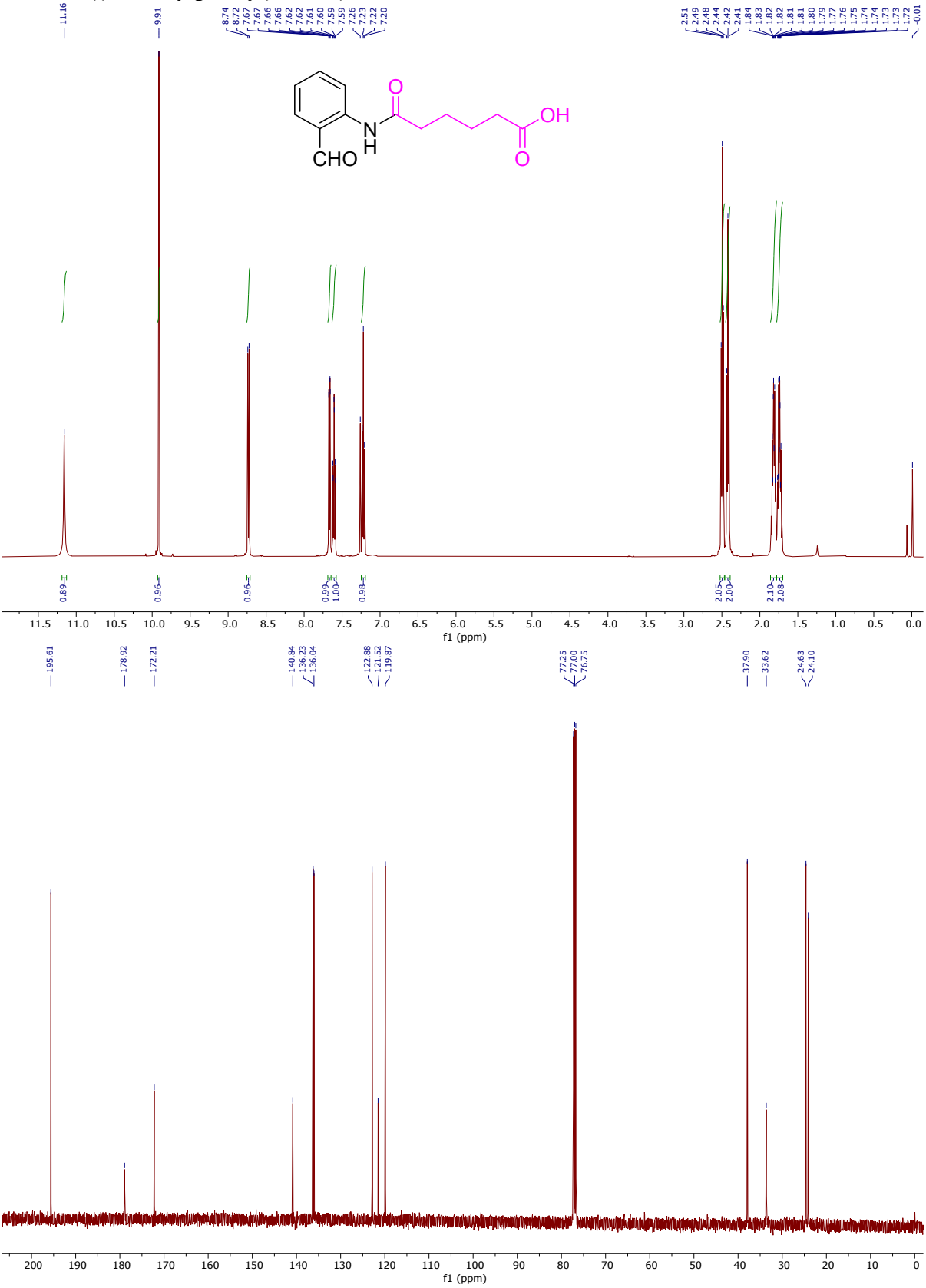
C-13: 2-((2-formylphenyl)carbamoyl)benzoic acid



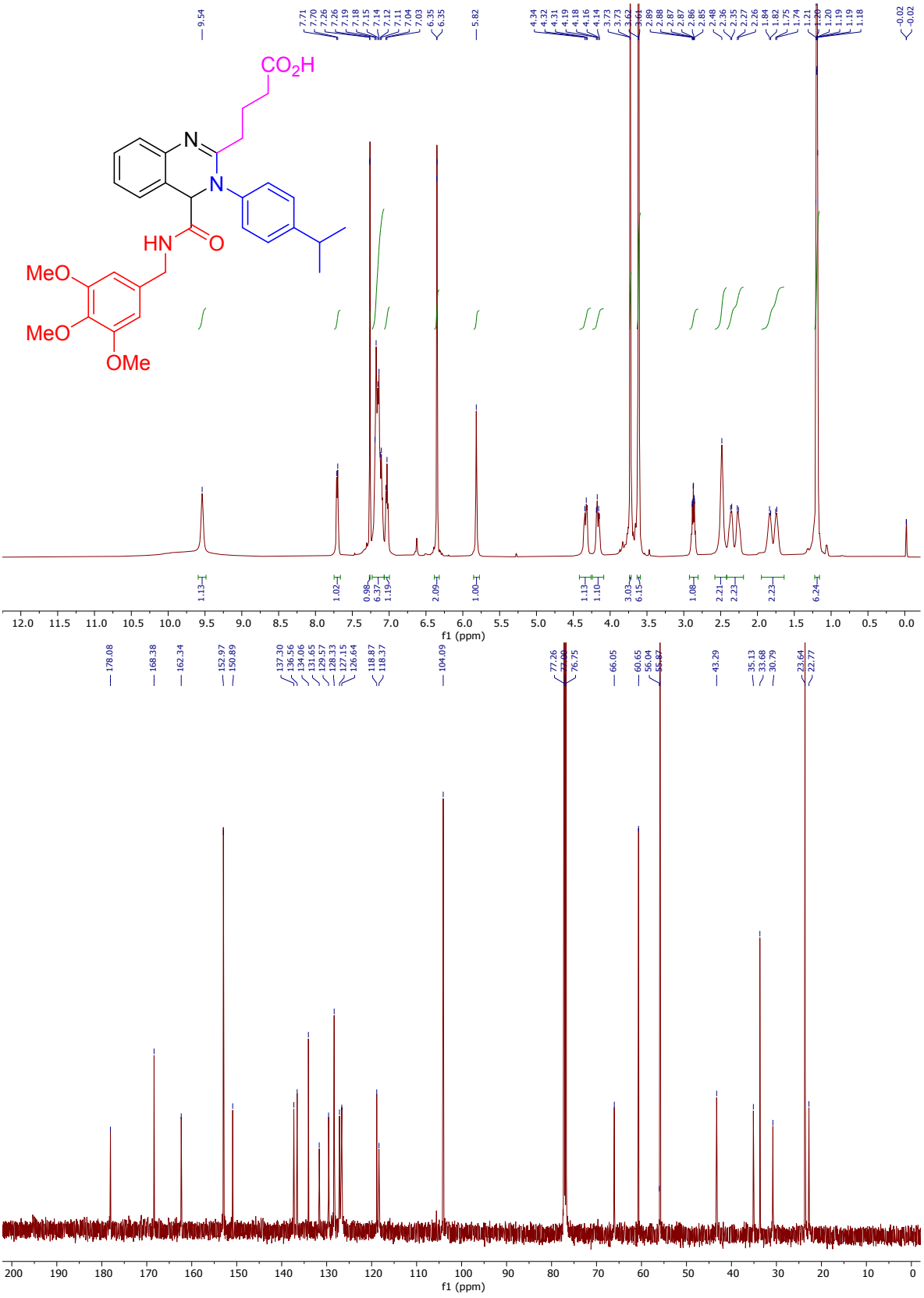
C-14: 2-((4-chloro-2-formylphenyl)carbamoyl)cyclopropane-1-carboxylic acid

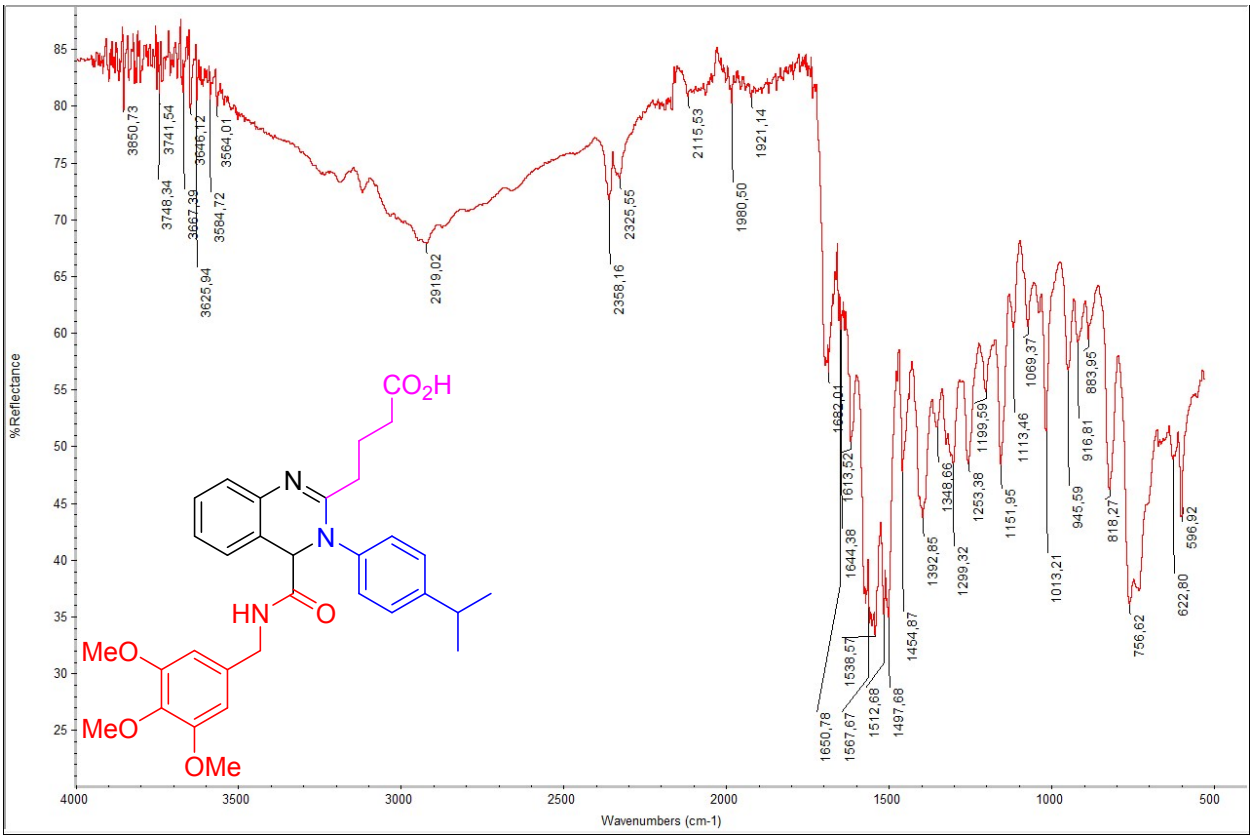


C-15: 6-((2-formylphenyl)amino)-6-oxohexanoic acid

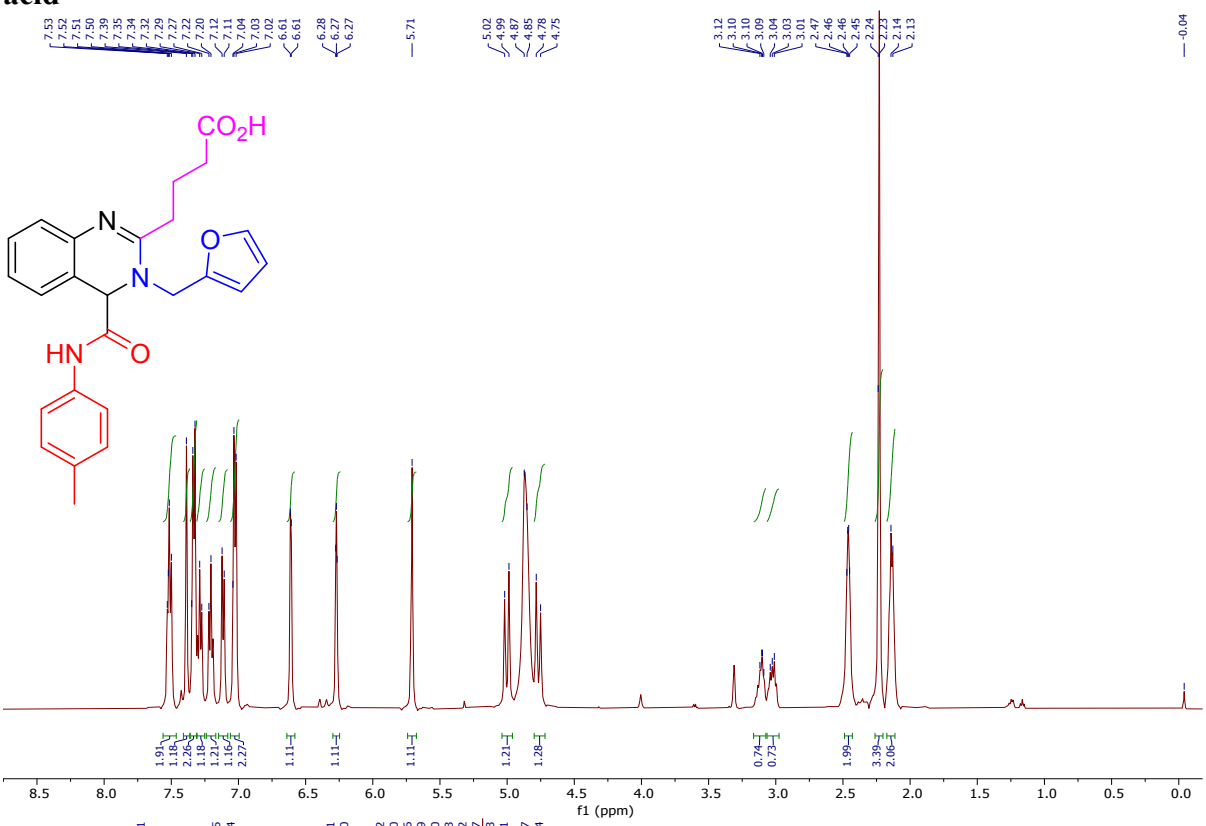


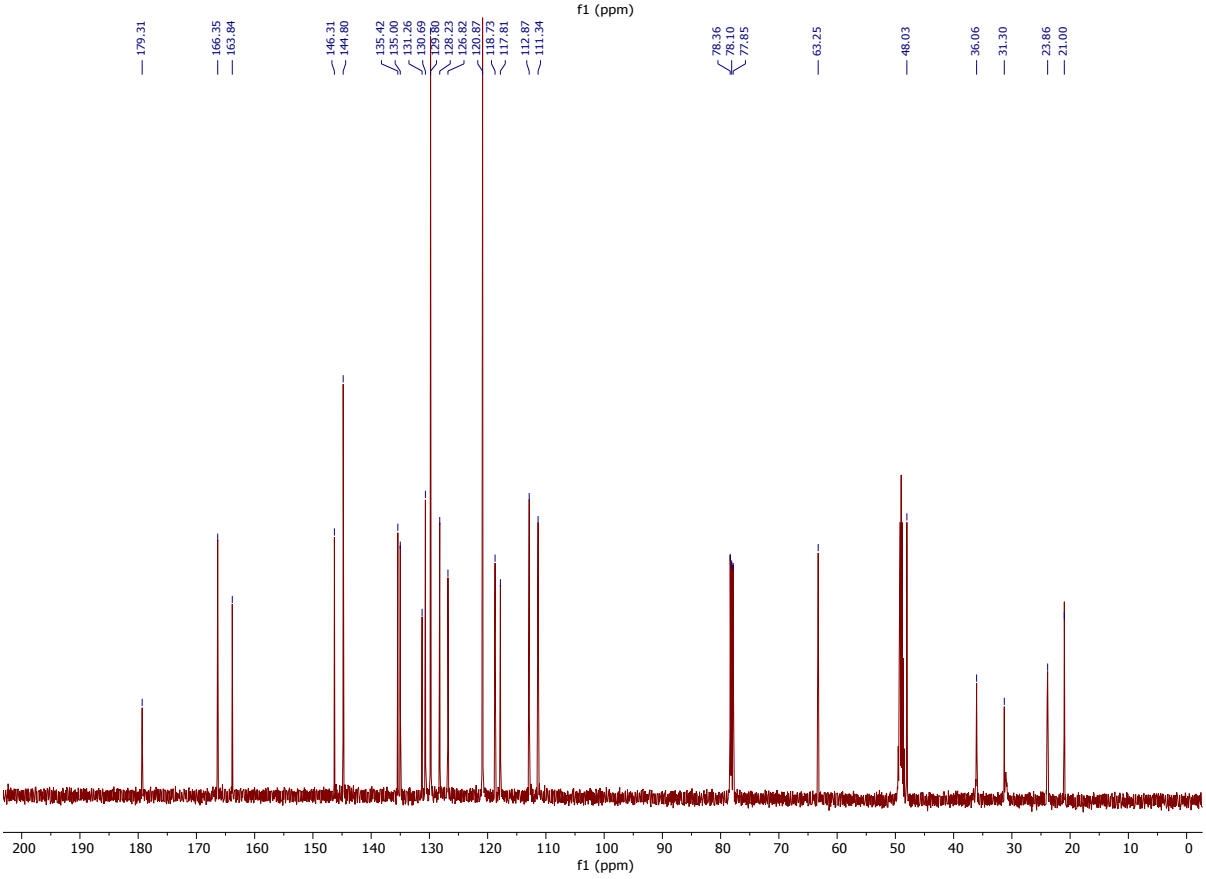
A24: 4-(3-(4-isopropylphenyl)-4-((3,4,5-trimethoxybenzyl)carbamoyl)-3,4-dihydroquinazolin-2-yl)butanoic acid:

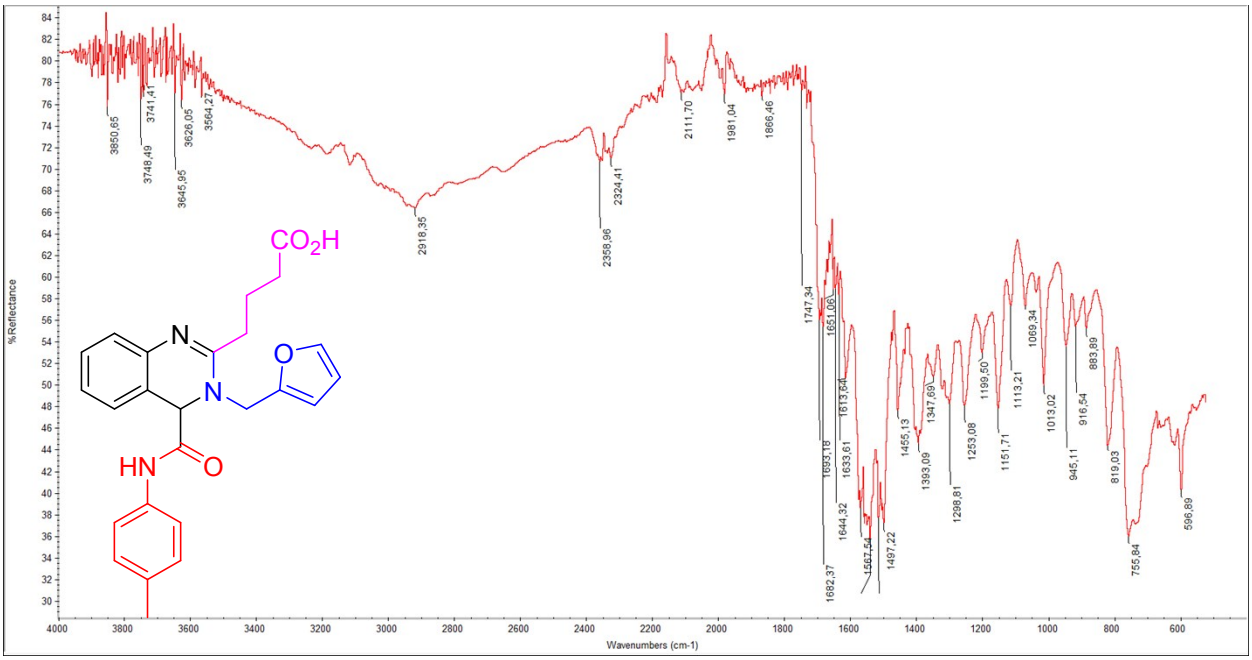



%Reflectance
Wavenumbers (cm-1)
3850,73
3748,34
3741,54
3667,39
3646,12
3625,94
3584,72
3564,01
2919,02
2358,16
2325,55
2115,53
1980,50
1921,14
1682,01
1650,78
1644,38
1613,52
1567,67
1538,57
1512,68
1497,68
1454,87
1392,85
1348,66
1299,32
1253,38
1199,59
1151,95
1113,46
1069,37
1013,21
945,59
916,81
883,95
818,27
756,62
622,80
596,92
CO2H
N
N
HN
O
MeO
MeO
OMe

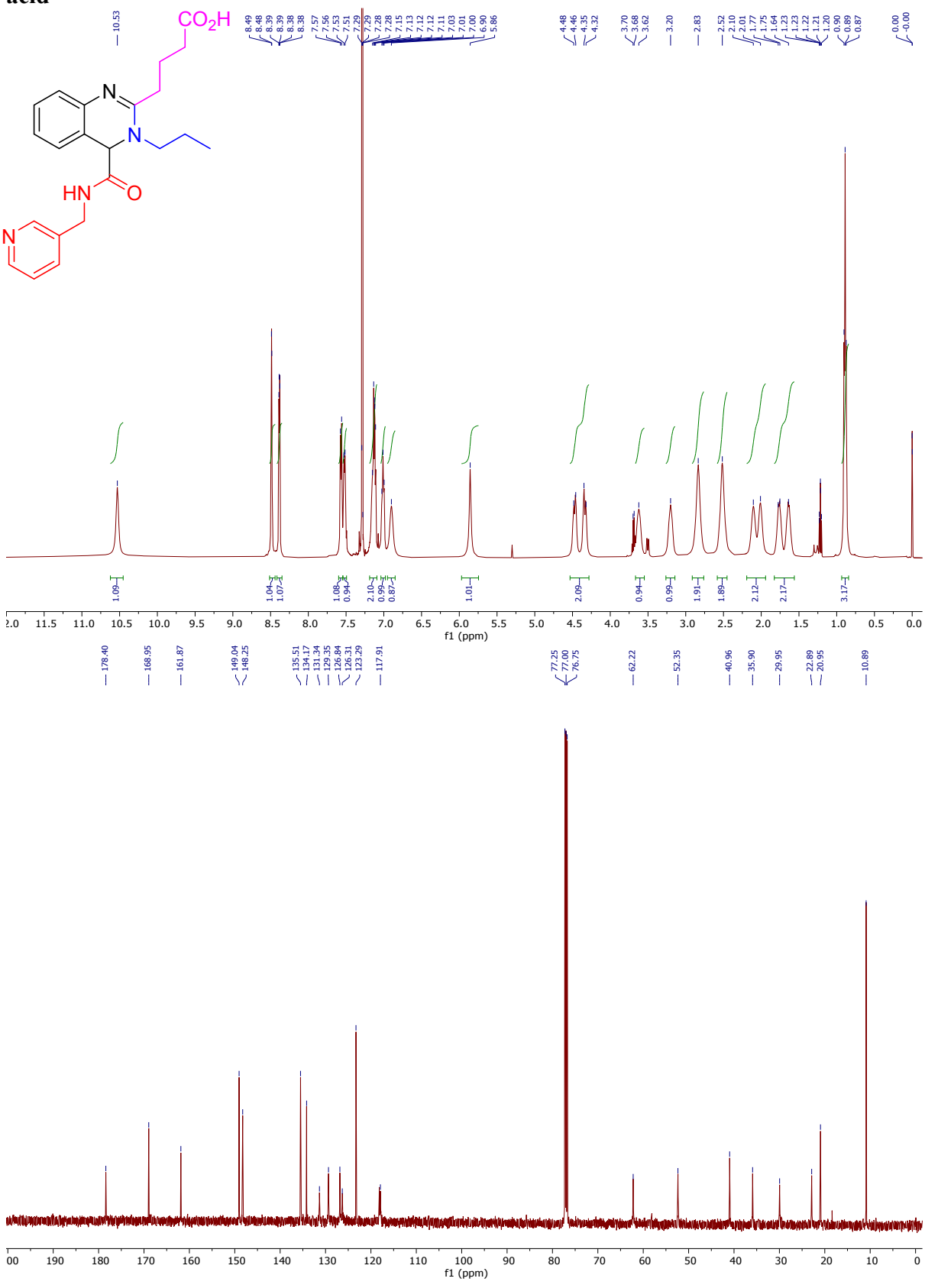
B13: 4-(3-(furan-2-ylmethyl)-4-(p-tolylcarbamoyl)-3,4-dihydroquinazolin-2-yl)butanoic acid

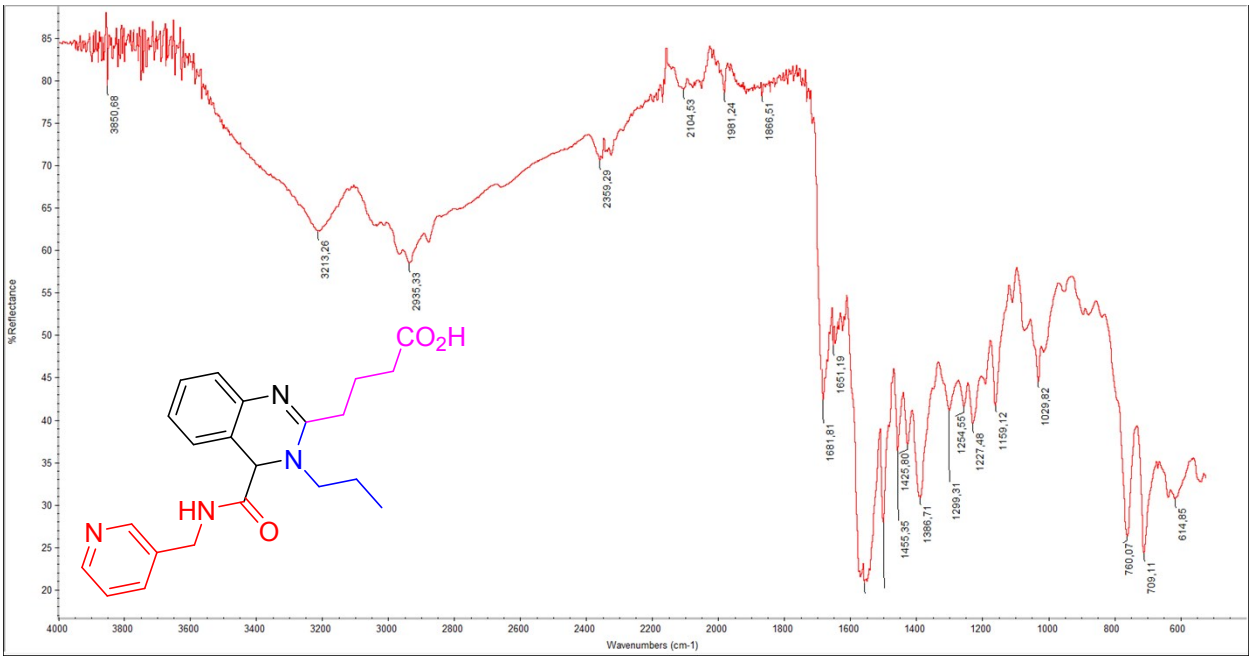




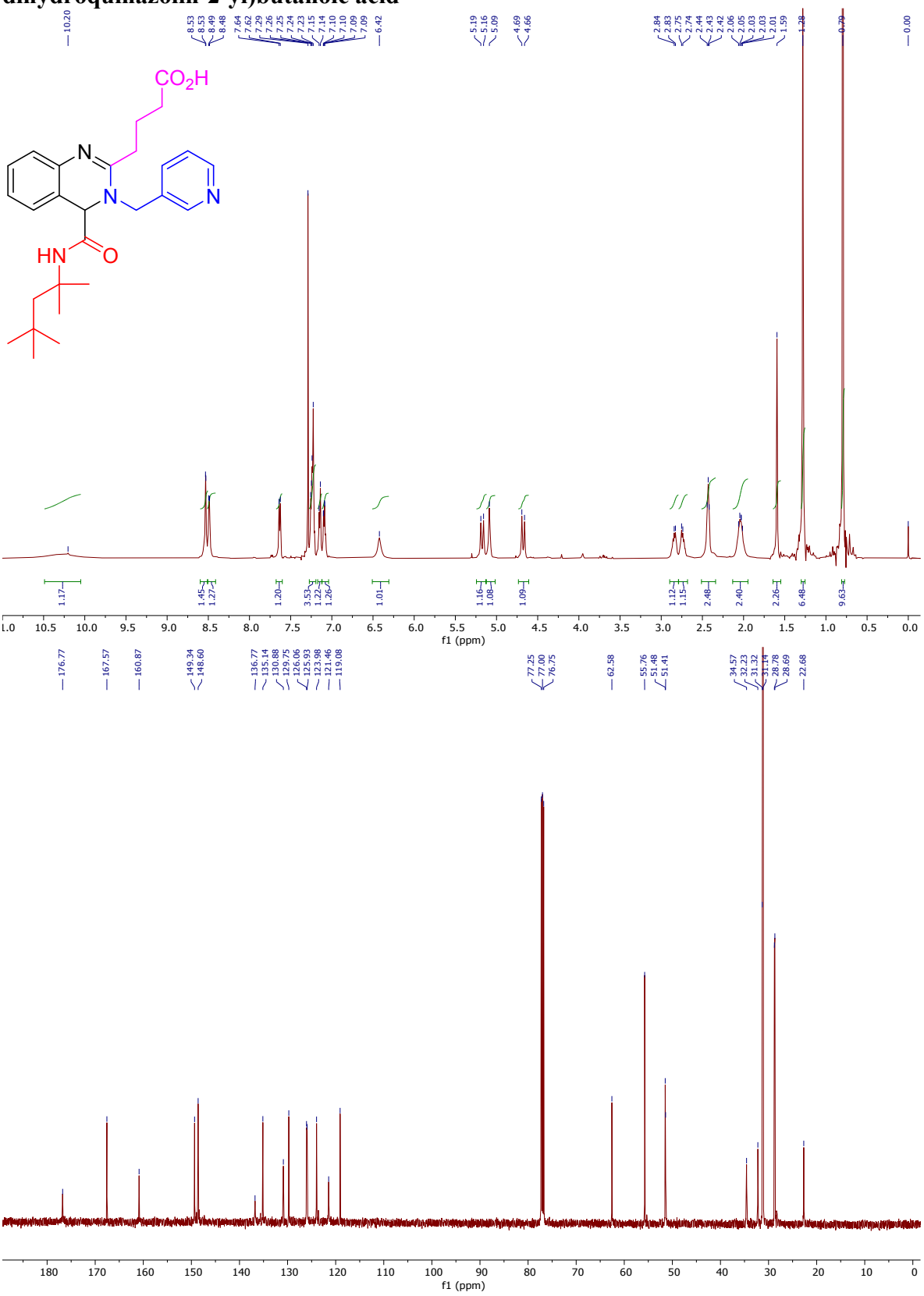

%Reflectance
Wavenumbers (cm-1)
3850,65
3748,49
3741,41
3645,95
3626,05
3564,27
2918,35
2358,96
2324,41
2111,70
1981,04
1866,46
1747,34
1682,37
1693,18
1651,06
1644,32
1633,61
1813,54
1567,54
1497,22
1455,13
1393,09
1347,69
1298,81
1253,08
1199,50
1151,71
1113,21
1069,34
1013,02
945,11
916,54
883,89
819,03
755,84
596,89
CO2H
N
N
O
HN
O

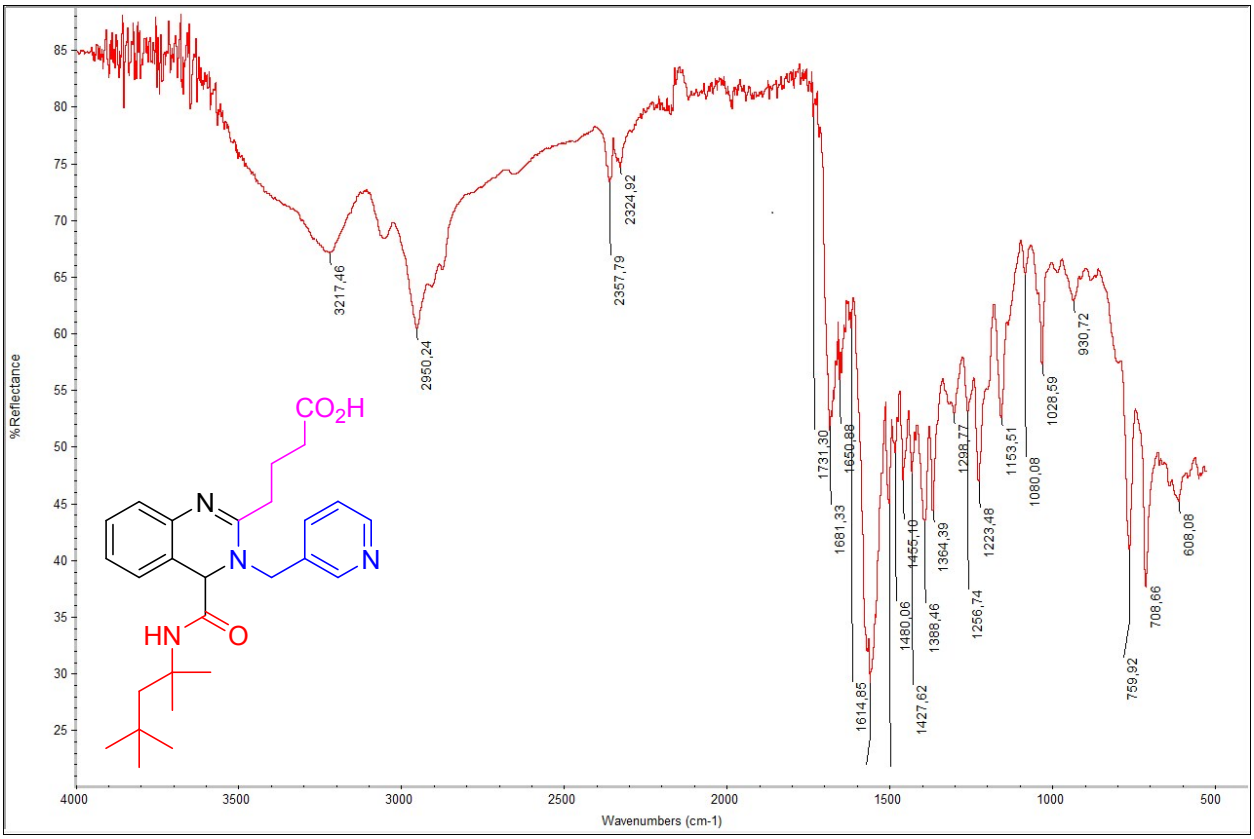
B20:4-(4-(benzylcarbamoyl)-3-(4-methoxyphenethyl)-3,4-dihydroquinazolin-2-yl)butanoic acid



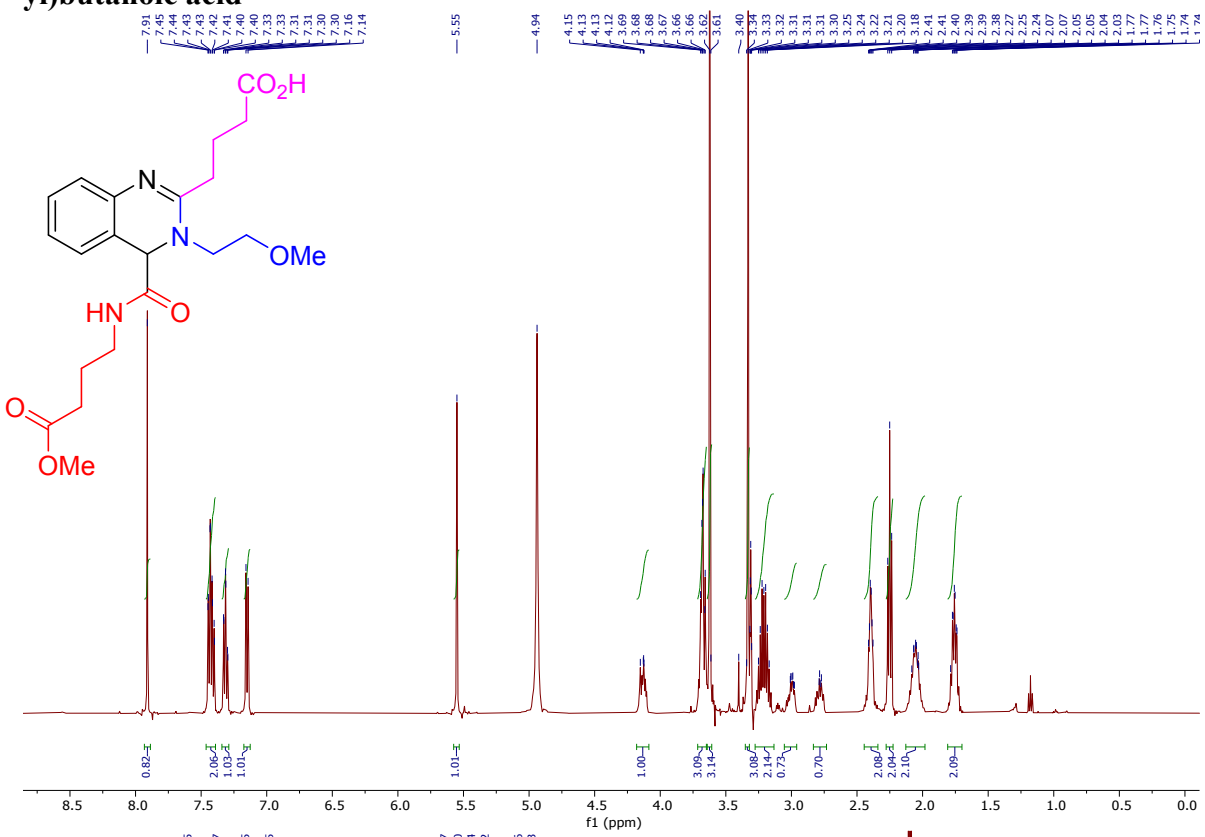

%Reflectance
Wavenumbers (cm-1)
3850,68
3213,26
2935,33
2359,29
2104,53
1981,24
1866,51
1681,81
1651,19
1455,35
1425,80
1386,71
1299,31
1254,55
1227,48
1159,12
1029,82
760,07
709,11
614,85
CO2H
N
N
HN
O
N

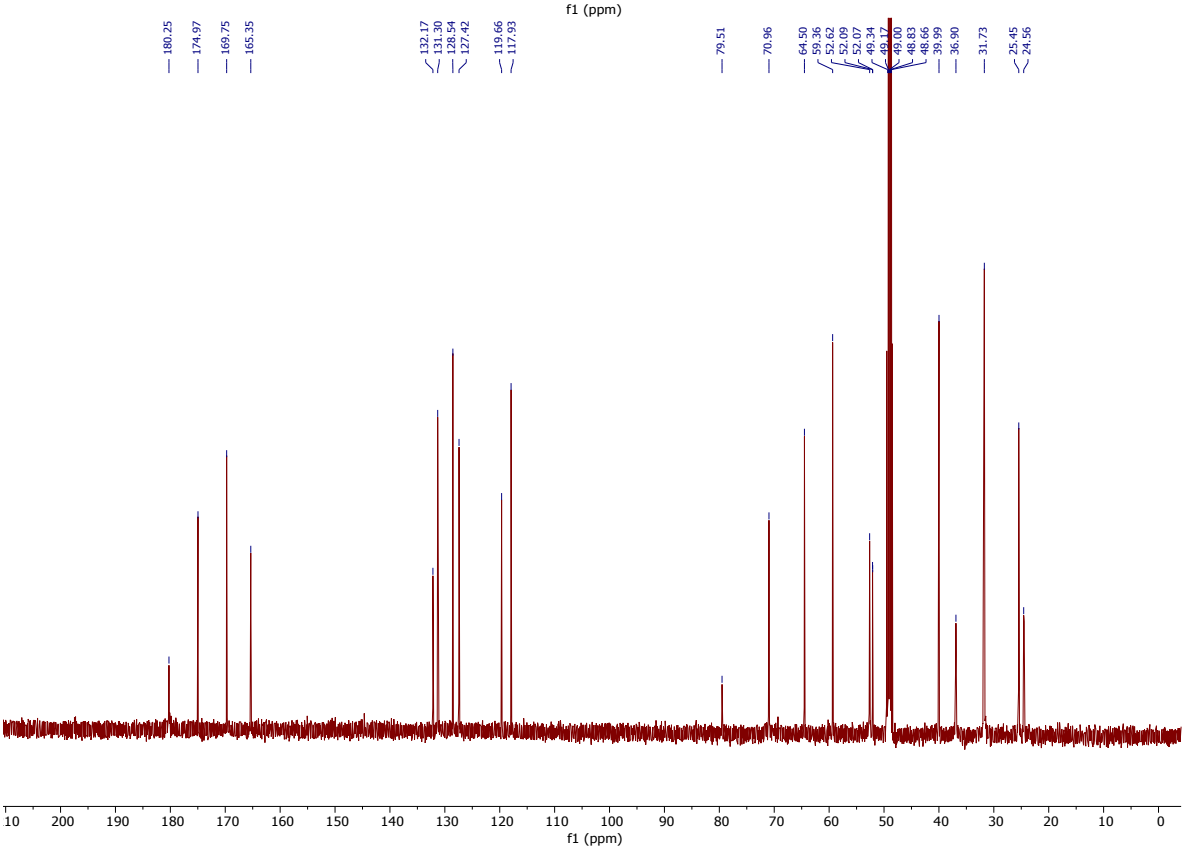
C1: 4-(3-(pyridin-3-ylmethyl)-4-((2,4,4-trimethylpentan-2-yl)carbamoyl)-3,4-dihydroquinazolin-2-yl)butanoic acid

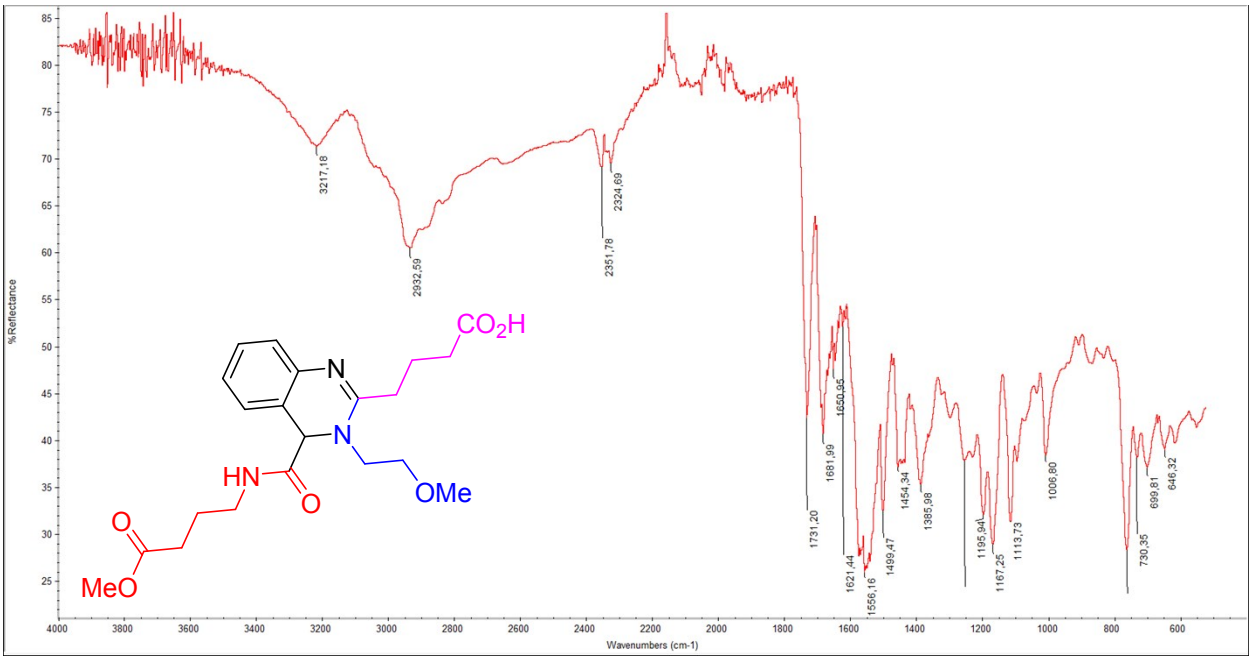



%Reflectance
Wavenumbers (cm-1)
3217,46
2950,24
2357,79
2324,92
1731,30
1681,33
1650,98
1614,85
1480,06
1455,10
1427,62
1388,46
1364,39
1298,77
1256,74
1223,48
1153,51
1080,08
1028,59
930,72
759,92
708,66
608,08
CO2H
N
N
N
HN
O

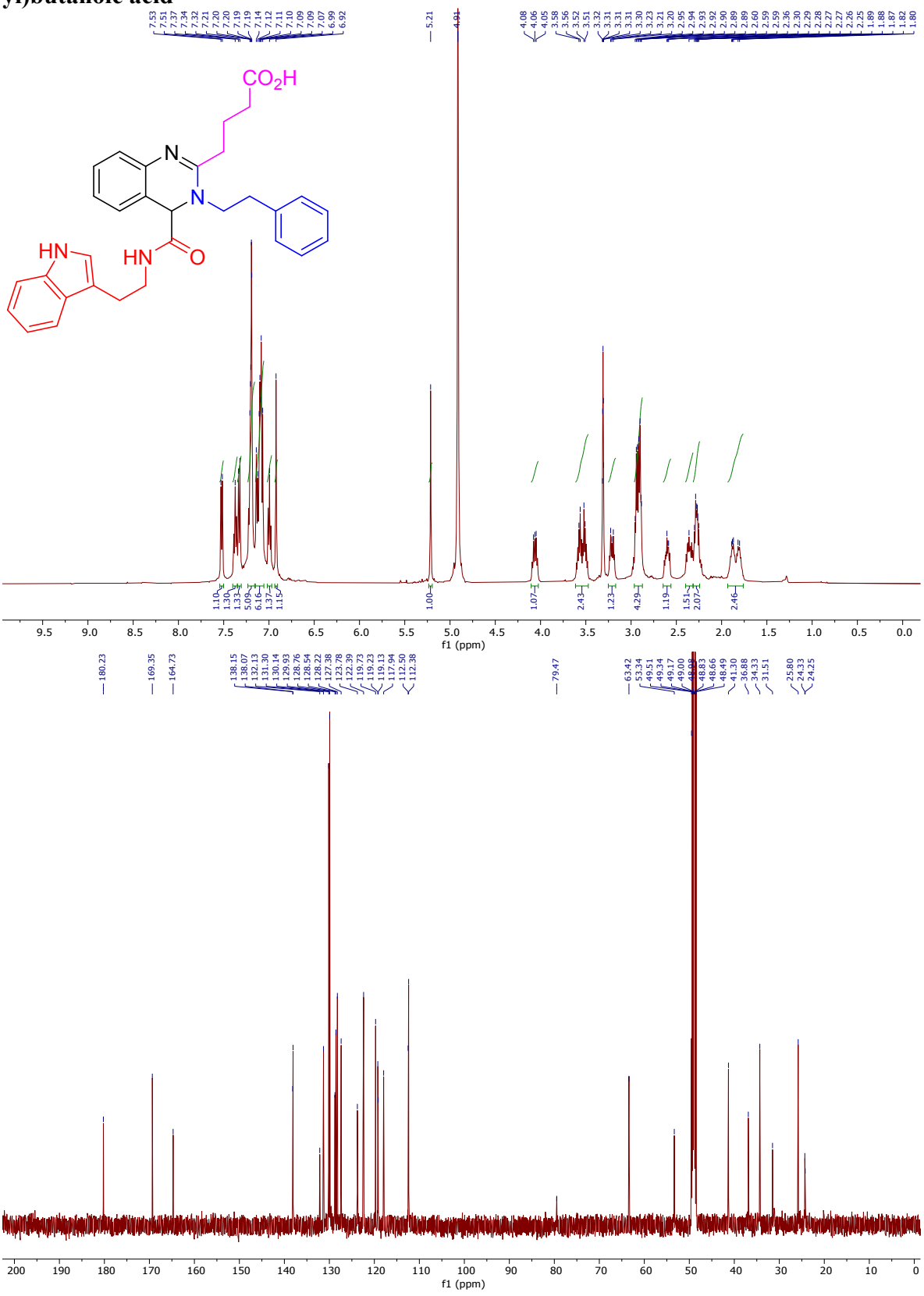
C11: 4-(4-((4-methoxy-4-oxobutyl)carbamoyl)-3-(2-methoxyethyl)-3,4-dihydroquinazolin-2-yl)butanoic acid

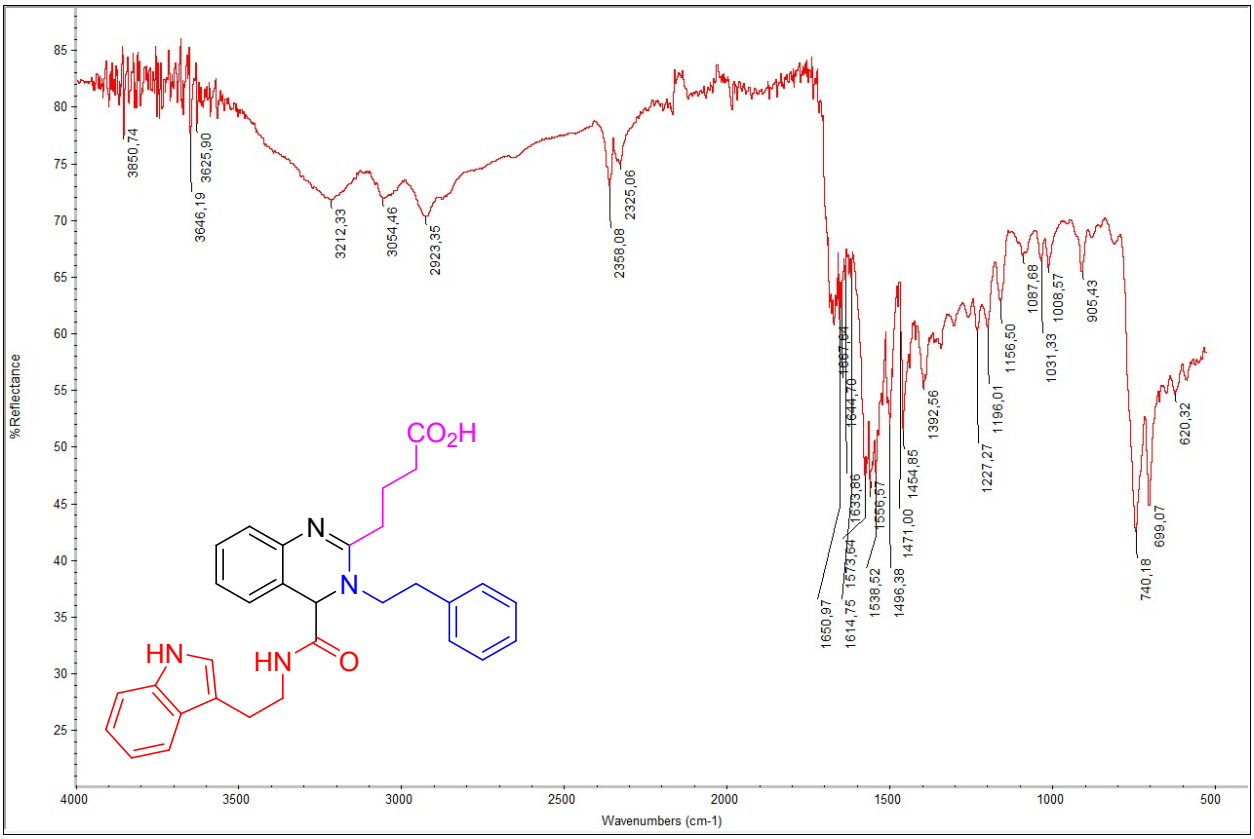




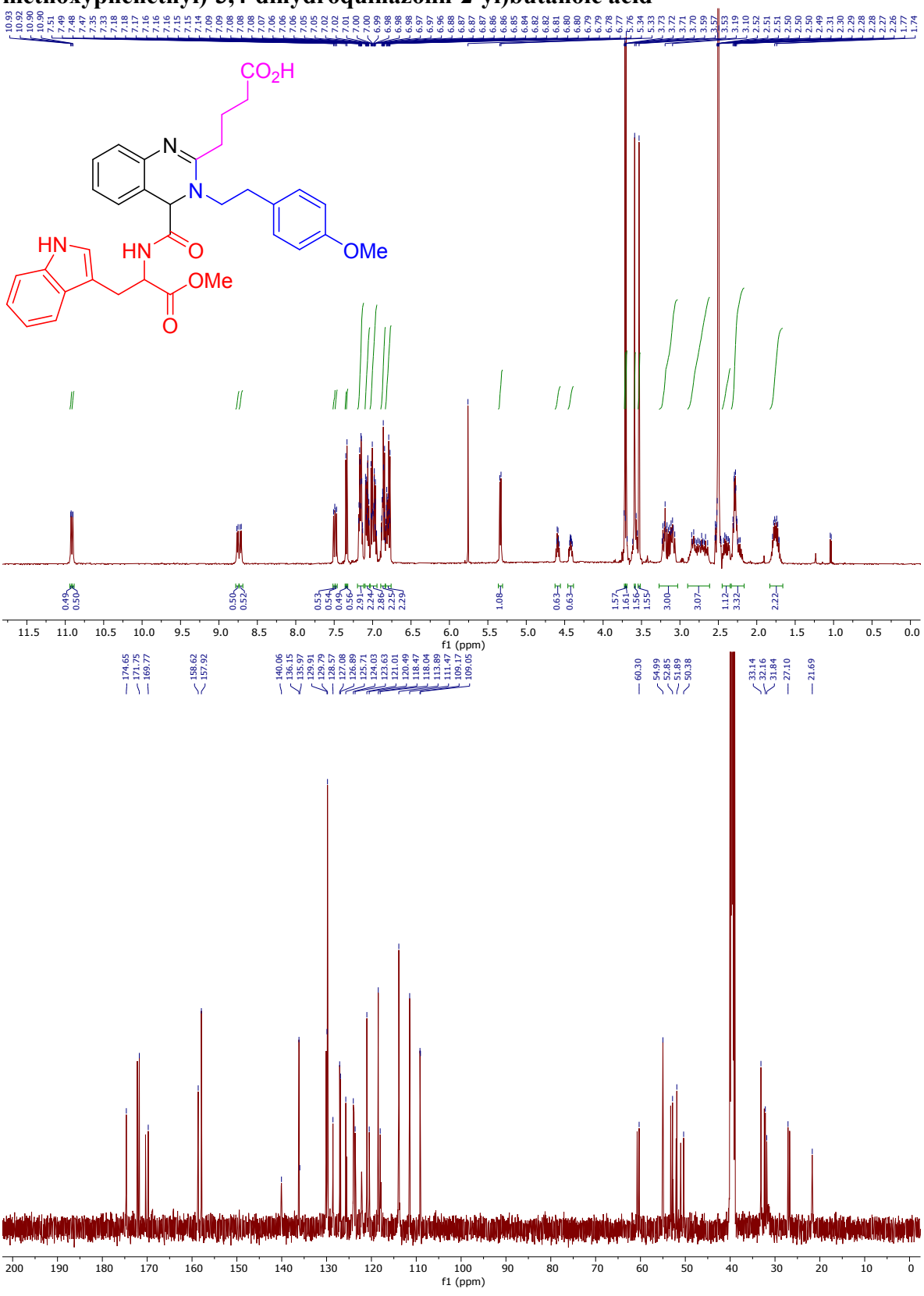

%Reflectance
Wavenumbers (cm-1)
3217,18
2932,59
2351,78
2324,69
1731,20
1681,99
1650,95
1621,44
1556,16
1499,47
1454,34
1385,98
1195,94
1167,25
1113,73
1006,80
730,35
699,81
646,32
CO2H
N
N
HN
O
O
OMe
O
MeO

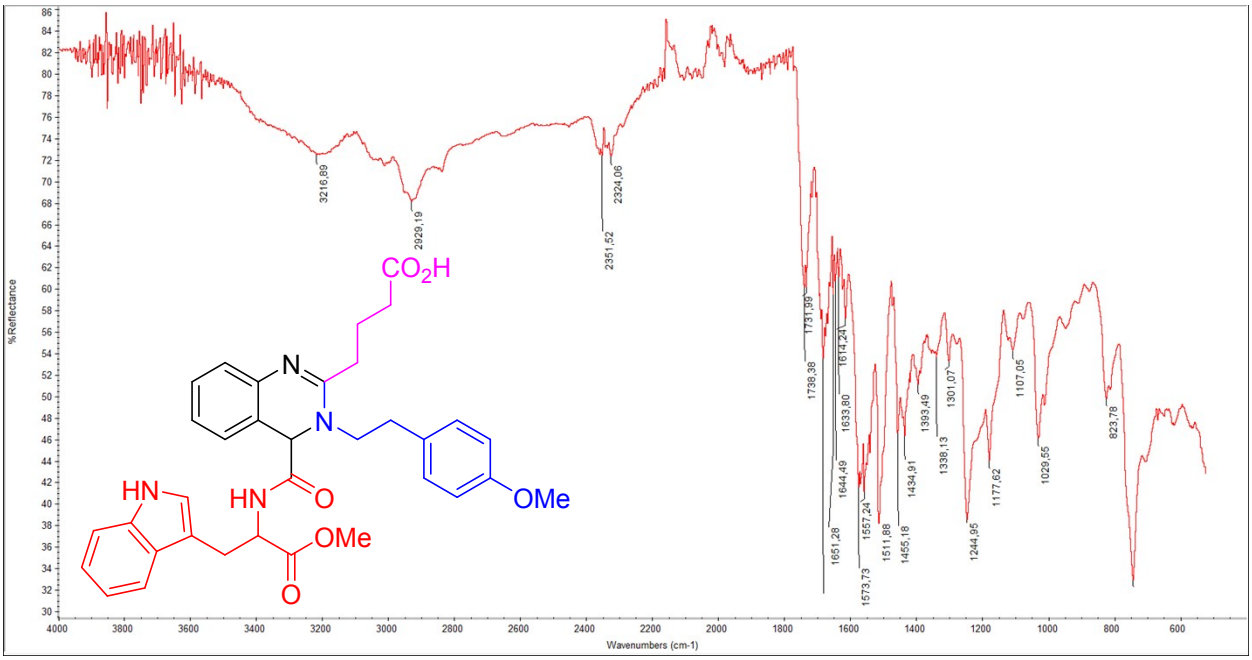
D8: 4-(4-((2-(1H-indol-3-yl)ethyl)carbamoyl)-3-phenethyl-3,4-dihydroquinazolin-2-yl)butanoic acid



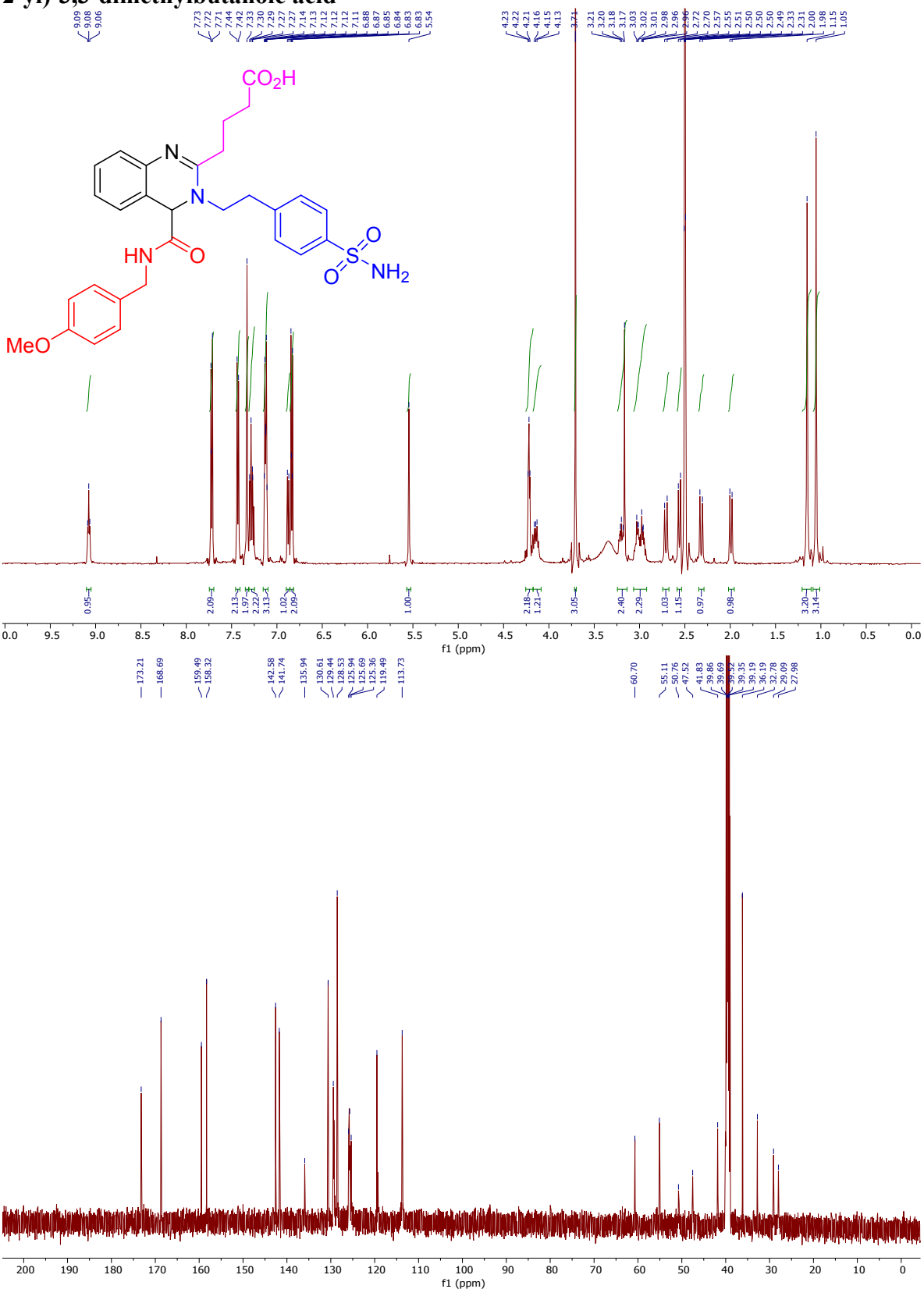

%Reflectance
Wavenumbers (cm-1)
3850,74
3625,90
3646,19
3212,33
3054,46
2923,35
2358,08
2325,06
1650,97
1614,75
1573,64
1538,52
1496,38
1471,00
1454,85
1392,56
1227,27
1196,01
1156,50
1087,68
1031,33
1008,57
905,43
740,18
699,07
620,32
CO2H
N
N
HN
HN
O

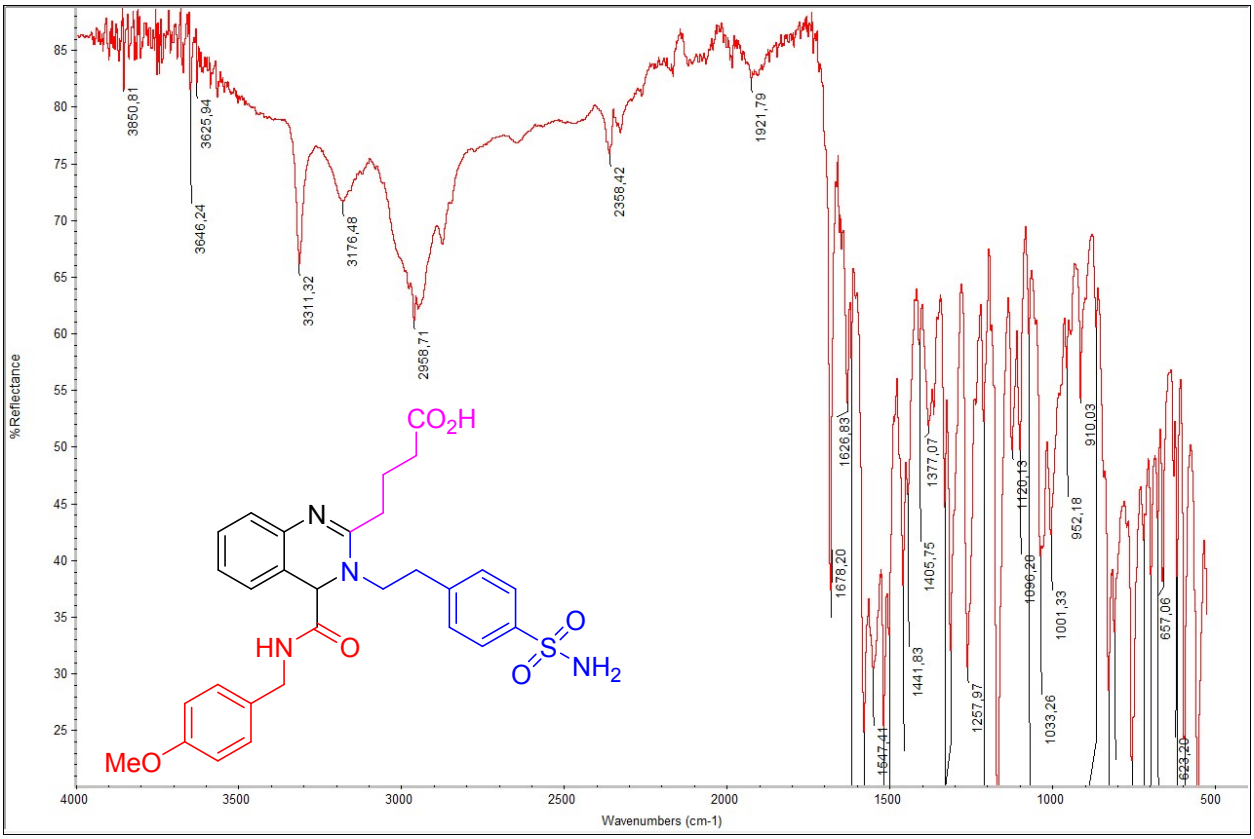
F2:4-(4-((3-(1H-indol-3-yl)-1-methoxy-1-oxopropan-2-yl)carbamoyl)-3-(4-methoxyphenethyl)-3,4-dihydroquinazolin-2-yl)butanoic acid



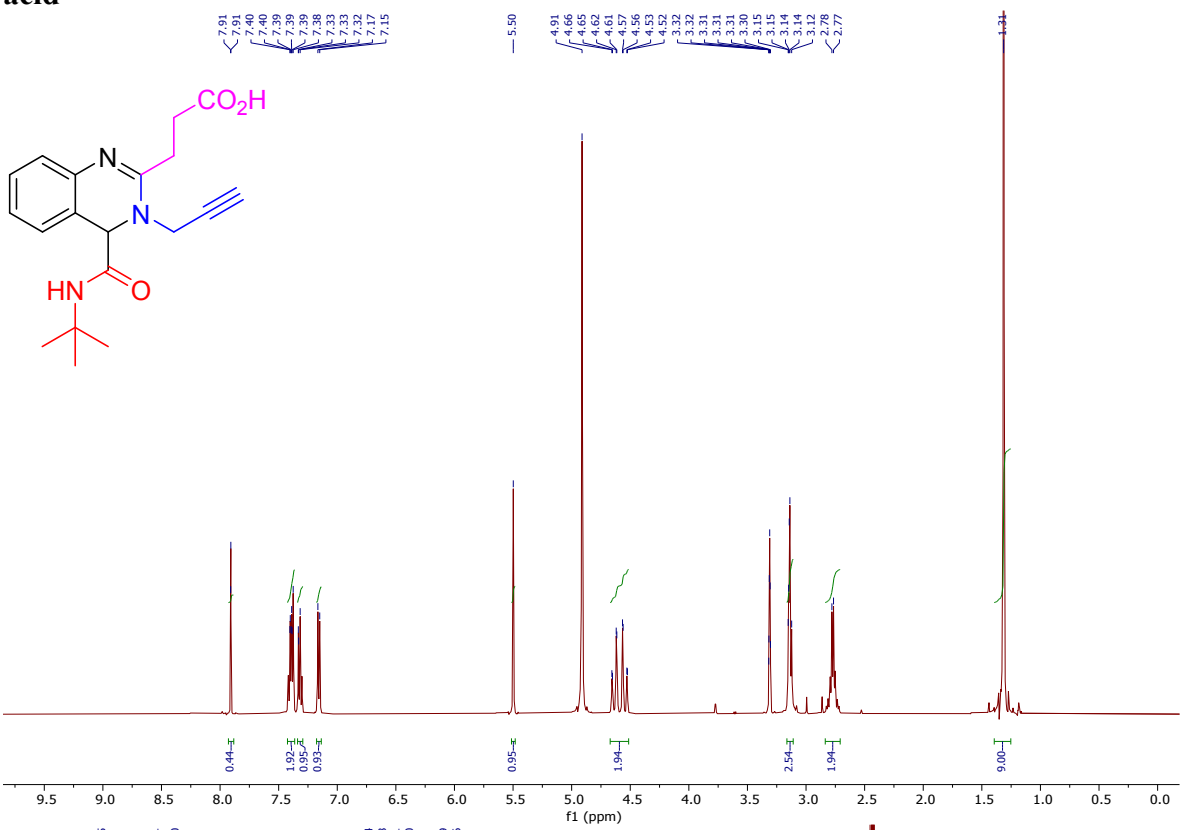

%Reflectance
Wavenumbers (cm-1)
3216,69
2929,19
2351,52
2324,06
1731,99
1738,38
1614,24
1633,80
1644,49
1651,28
1557,24
1573,73
1511,86
1455,18
1434,91
1393,49
1338,13
1301,07
1244,95
1177,62
1107,05
1029,55
823,78
CO2H
N
N
HN
HN
O
OMe
OMe
O

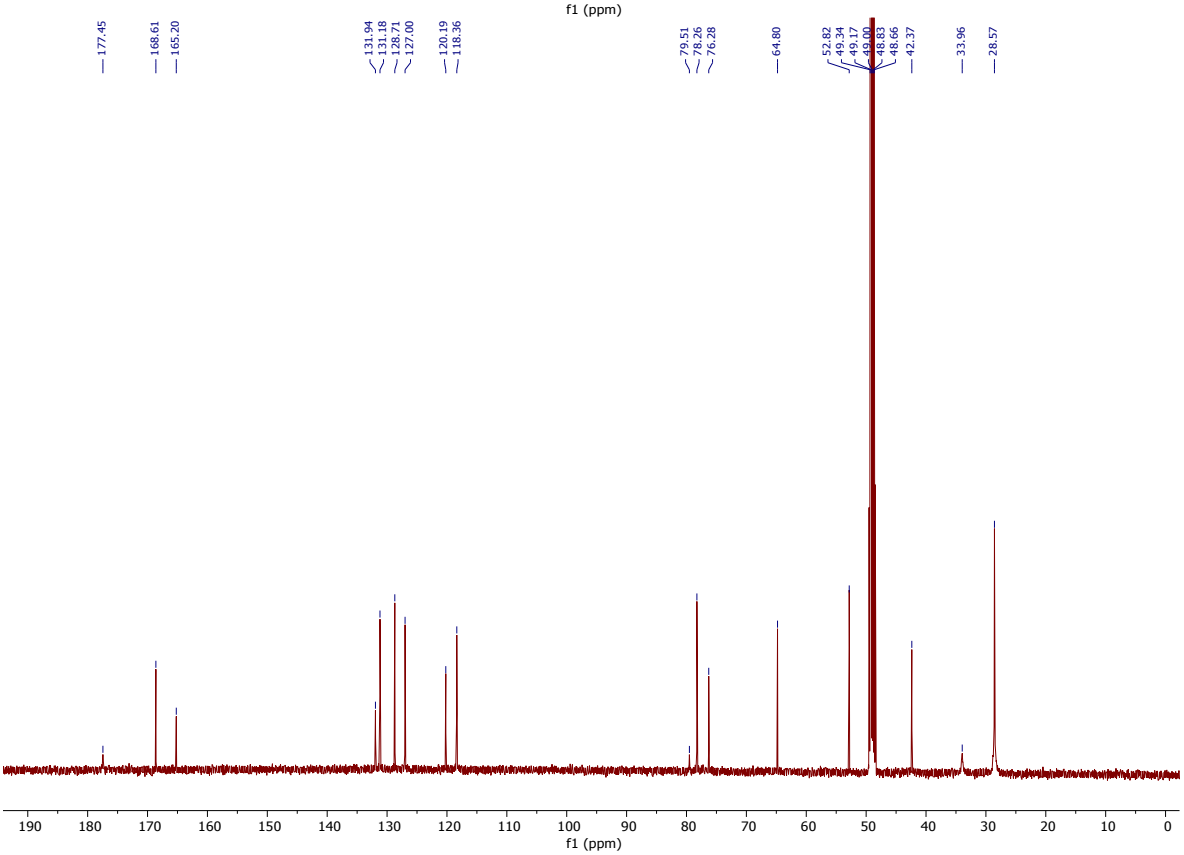
H13: 4-(4-((4-methoxybenzyl)carbamoyl)-3-(4-sulfamoylphenethyl)-3,4-dihydroquinazolin-2-yl)-3,3-dimethylbutanoic acid

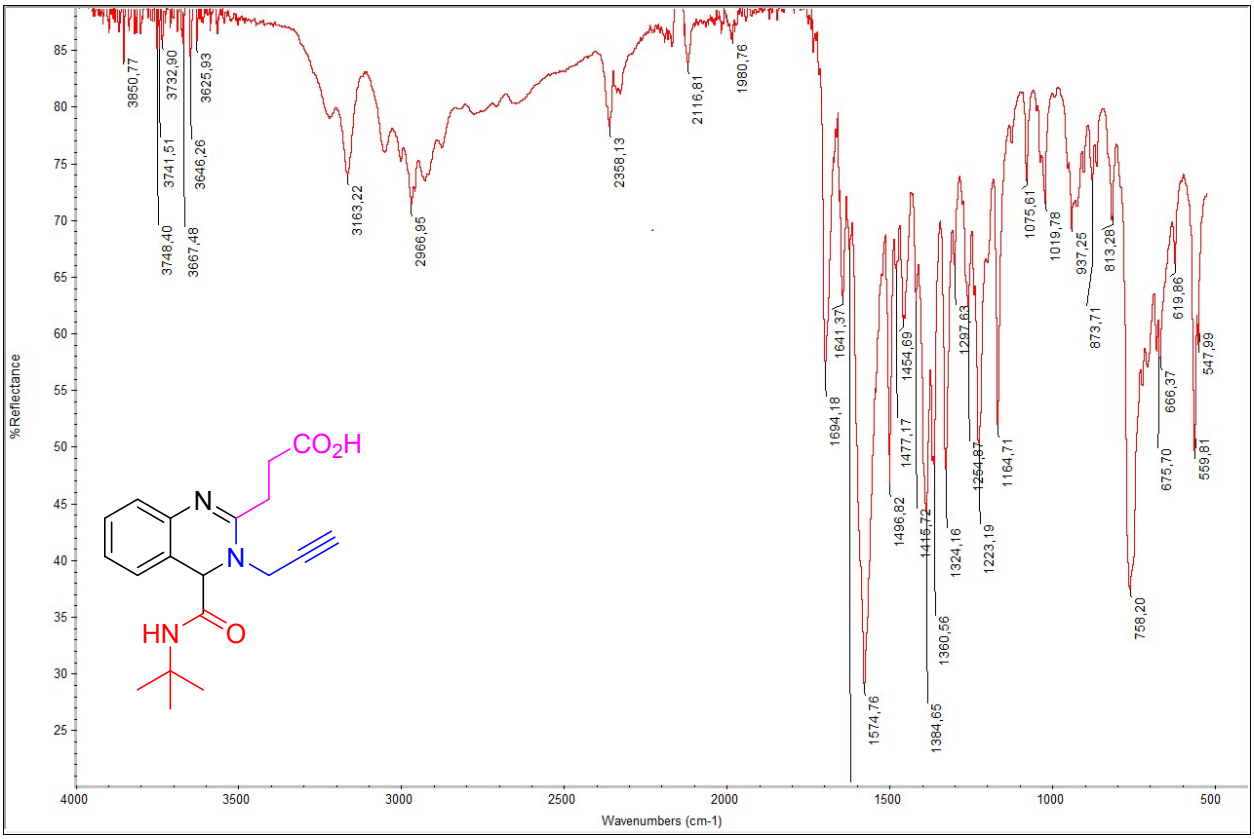



%Reflectance
Wavenumbers (cm-1)
3850,81
3625,94
3646,24
3311,32
3176,48
2958,71
2358,42
1921,79
1626,83
1678,20
1647,41
1441,83
1405,75
1377,07
1257,97
1120,13
1096,20
1033,26
1001,33
952,18
910,03
657,06
623,20
CO2H
N
N
HN
O
O
S
O
O
NH2
MeO

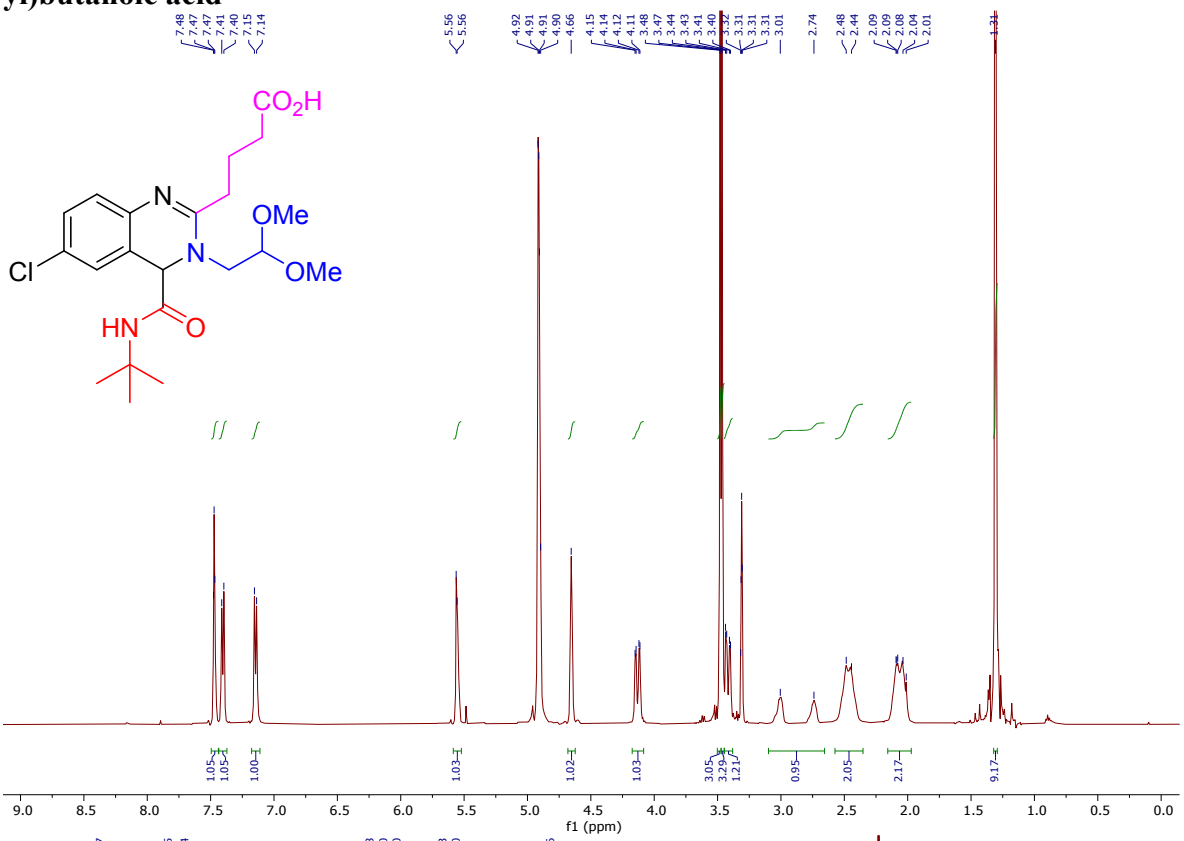
H17: 3-(4-(tert-butylcarbamoyl)-3-(prop-2-yn-1-yl)-3,4-dihydroquinazolin-2-yl)propanoic acid

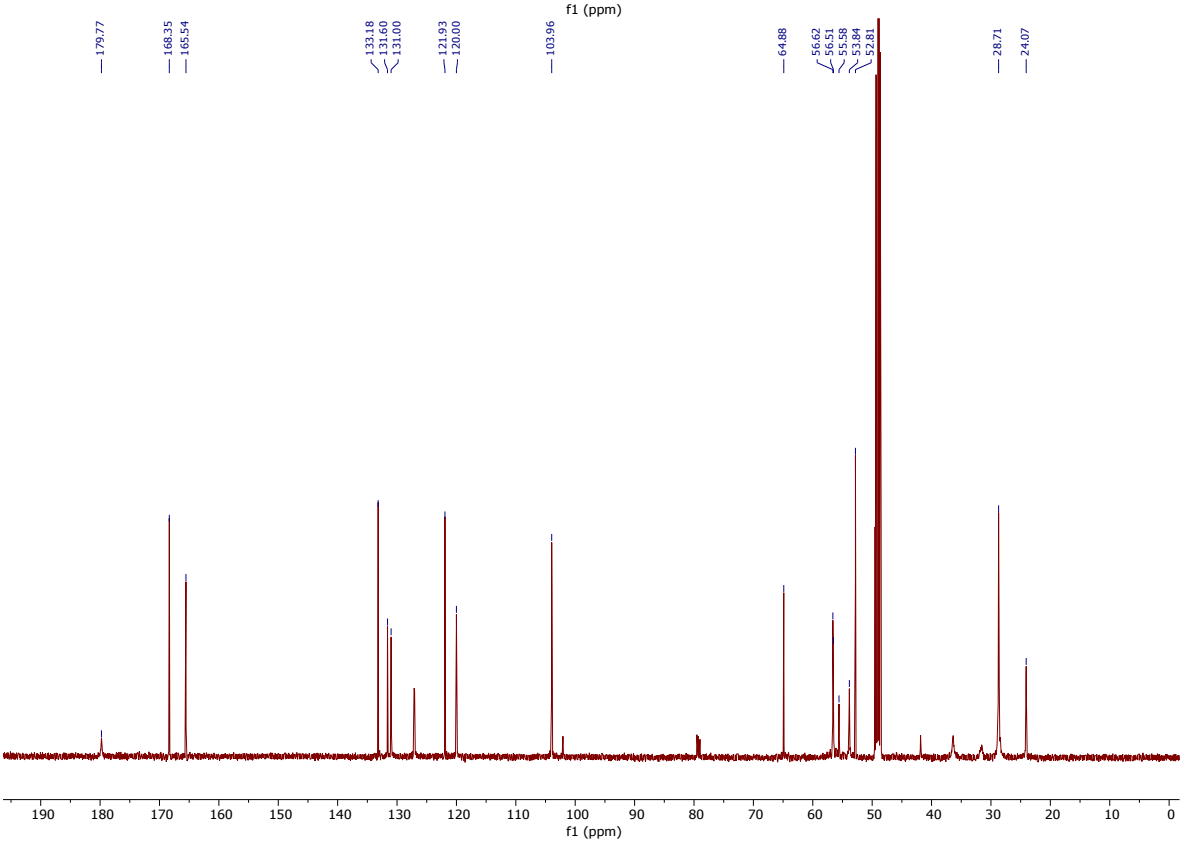


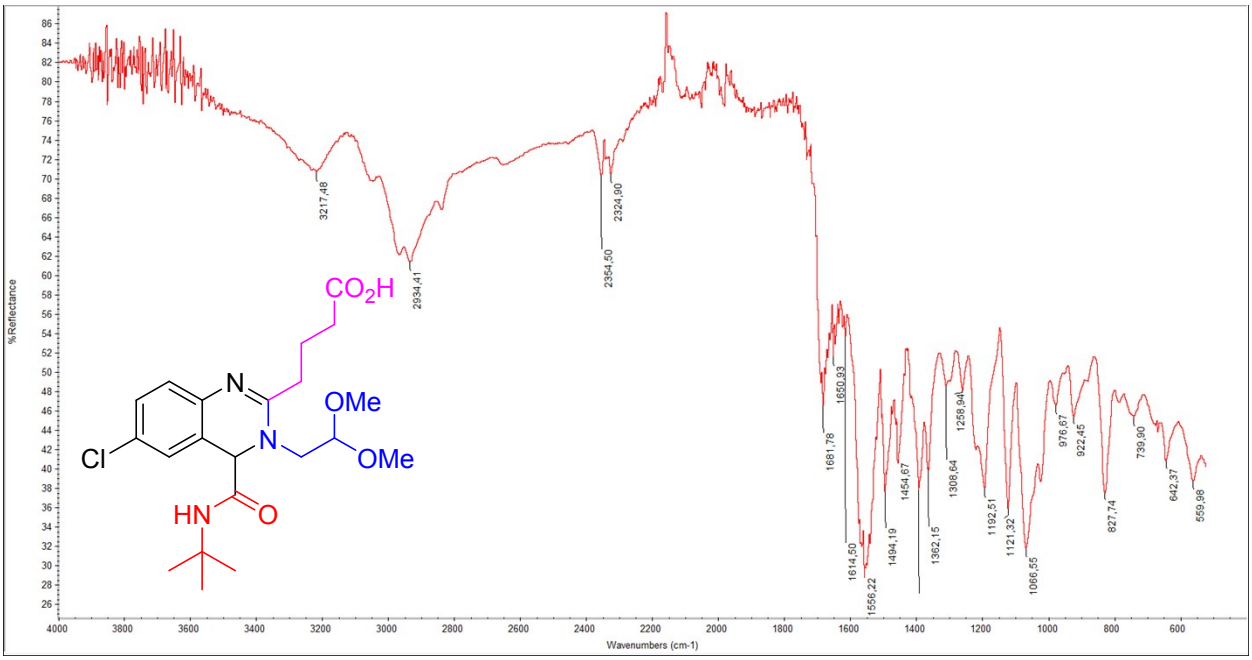



%Reflectance
Wavenumbers (cm-1)
3850,77
3741,51
3732,90
3646,26
3625,93
3748,40
3667,48
3163,22
2966,95
2358,13
2116,81
1980,76
1694,18
1641,37
1574,76
1496,82
1477,17
1454,69
1416,72
1384,65
1360,56
1324,16
1297,63
1264,87
1223,19
1164,71
1075,61
1019,78
937,25
873,71
813,28
758,20
675,70
666,37
619,86
559,81
547,99
CO2H
N
N
HN
O

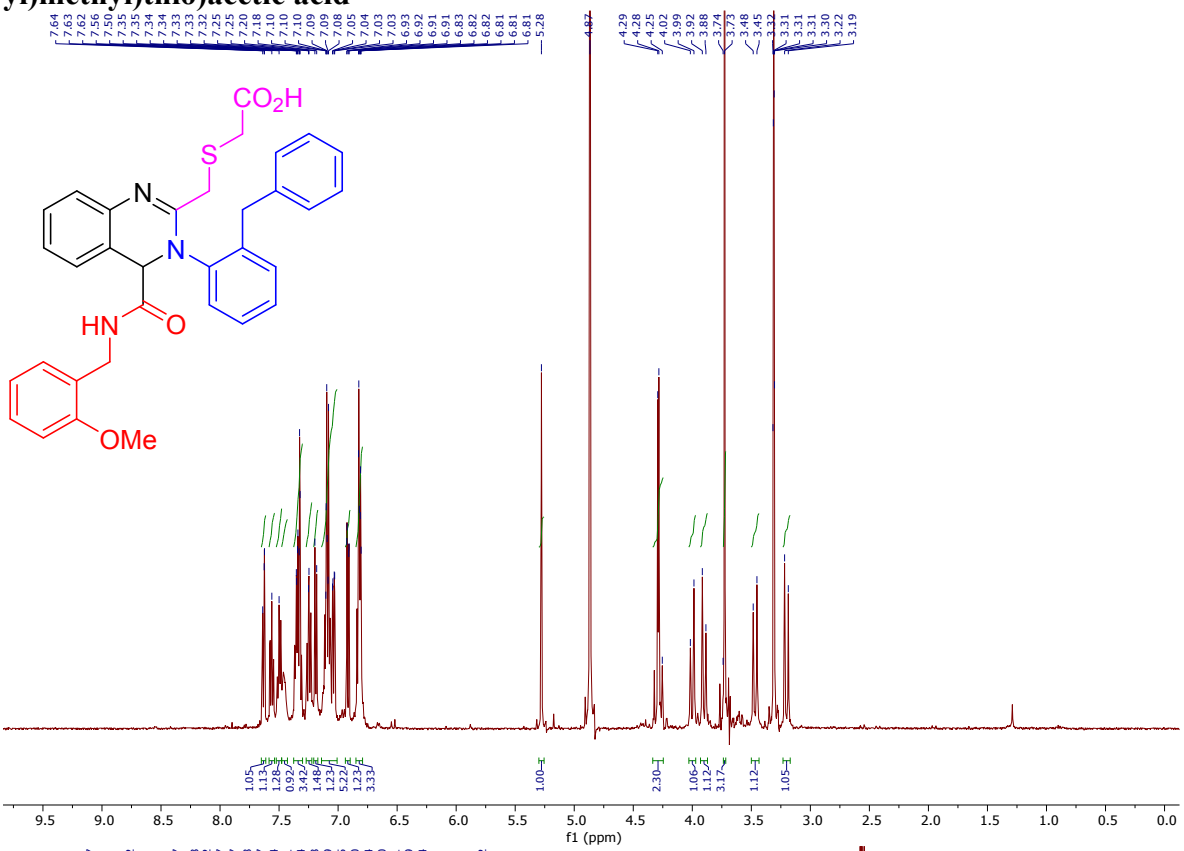
H21: 4-(4-(tert-butylcarbamoyl)-6-chloro-3-(2,2-dimethoxyethyl)-3,4-dihydroquinazolin-2-yl)butanoic acid

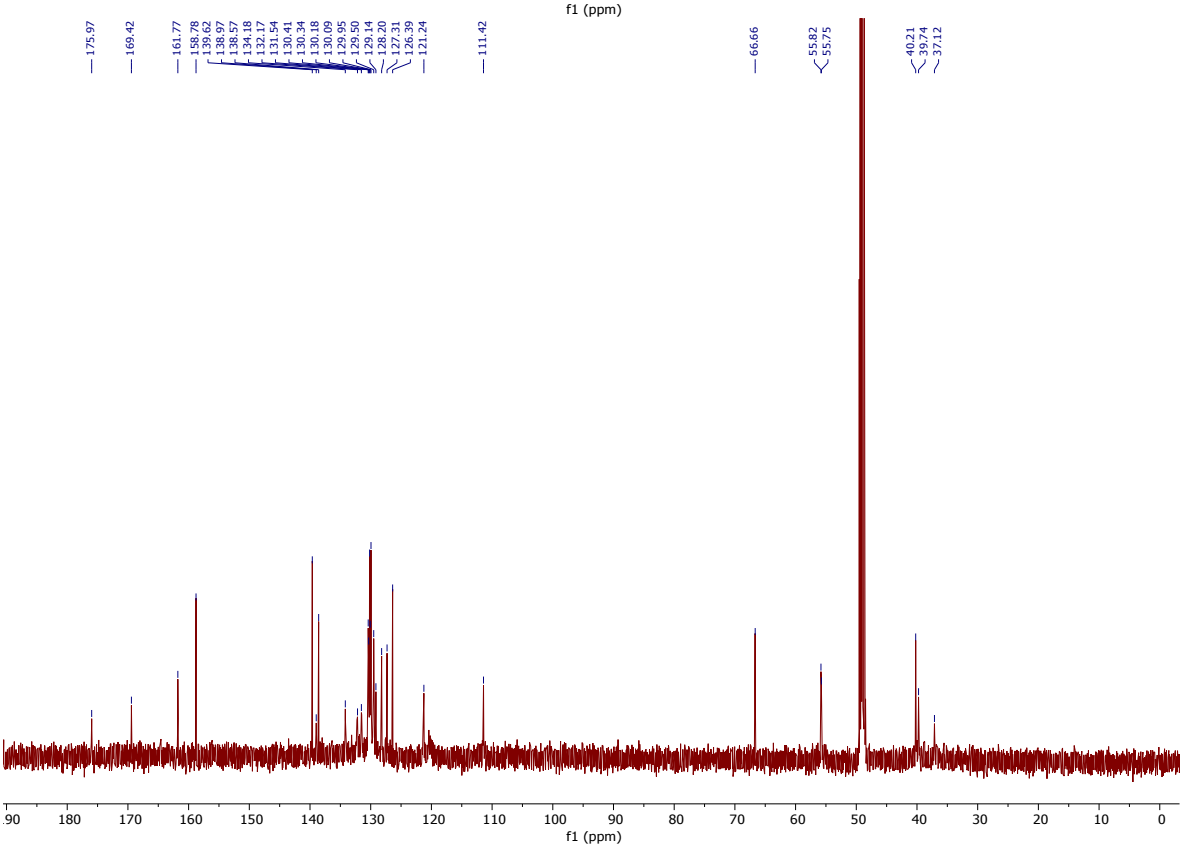


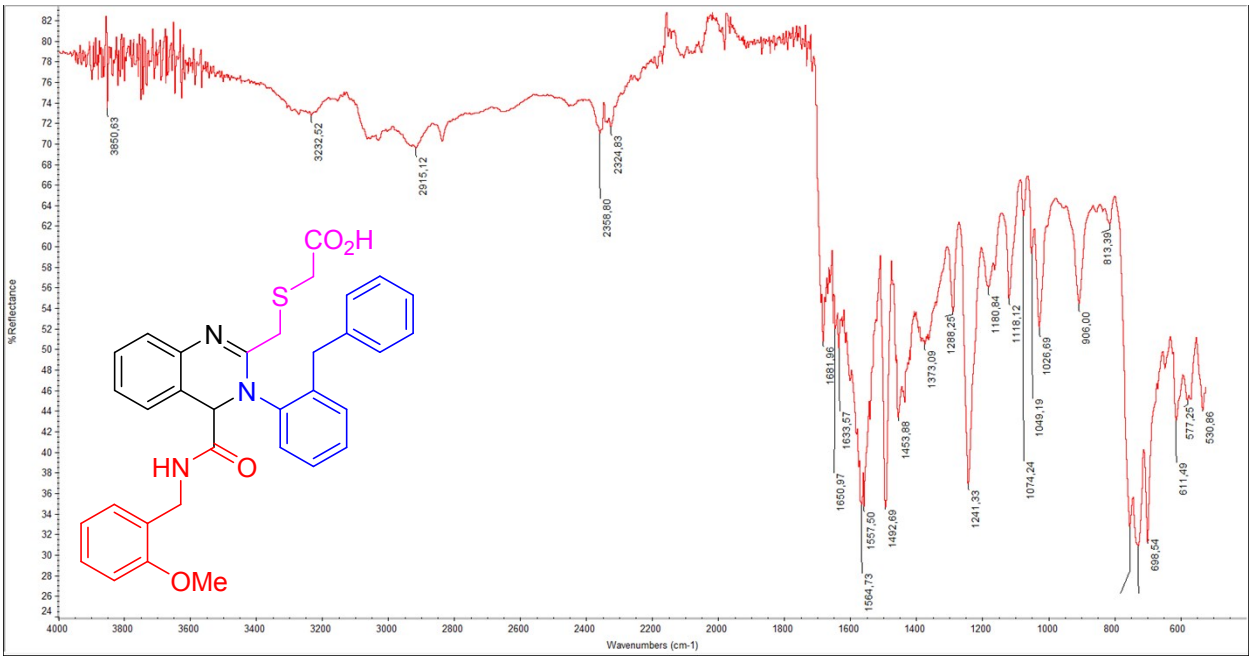



%Reflectance
Wavenumbers (cm-1)
3217,48
2934,41
2354,50
2324,90
1681,78
1650,93
1614,50
1556,22
1494,19
1454,67
1362,15
1308,64
1258,94
1192,51
1121,32
1066,55
976,67
922,45
827,74
739,90
642,37
559,98
CO2H
N
OMe
N
OMe
Cl
HN
O

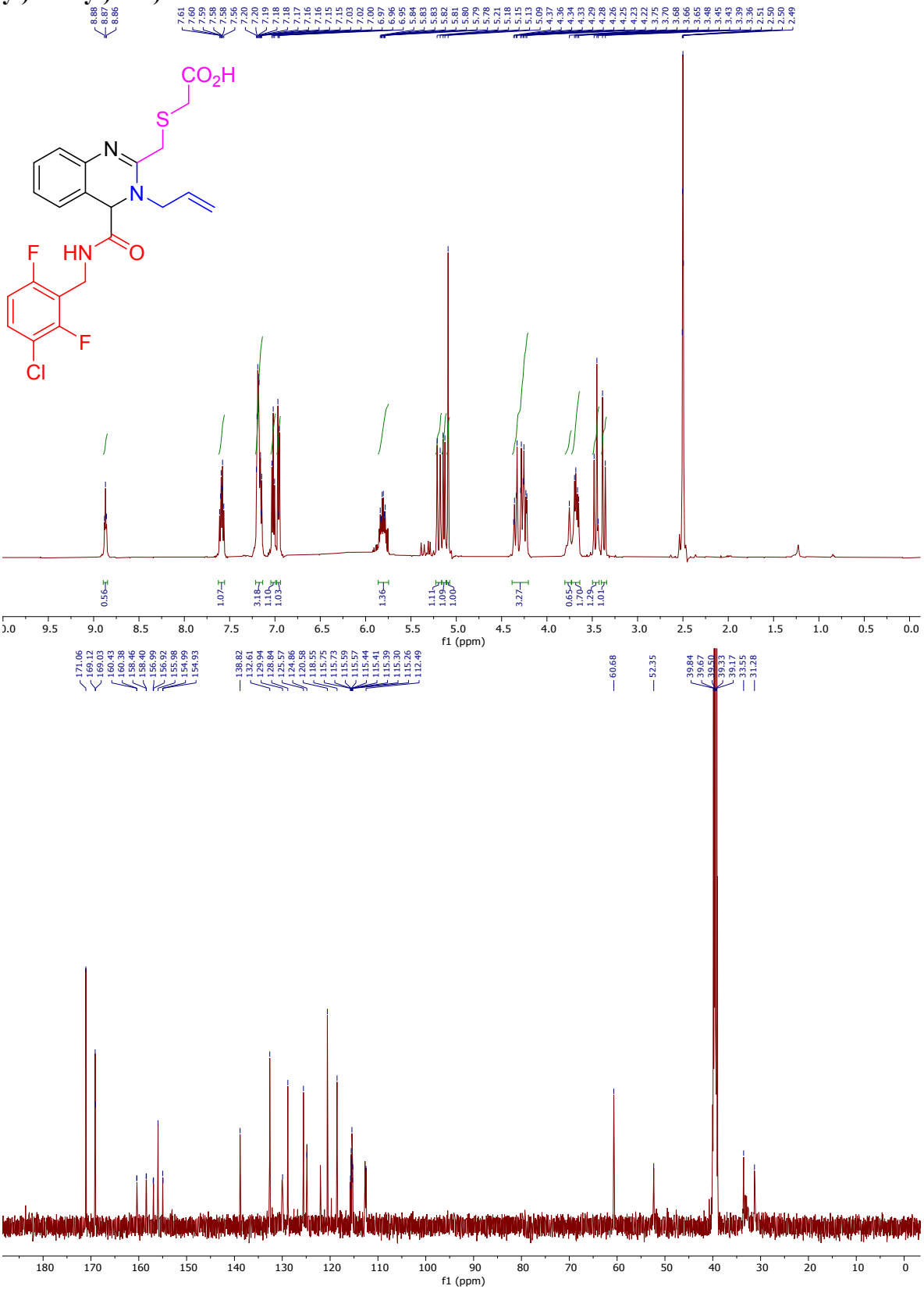
J12: 2-(((3-(2-benzylphenyl)-4-((2-methoxybenzyl)carbamoyl)-3,4-dihydroquinazolin-2-yl)methyl)thio)acetic acid

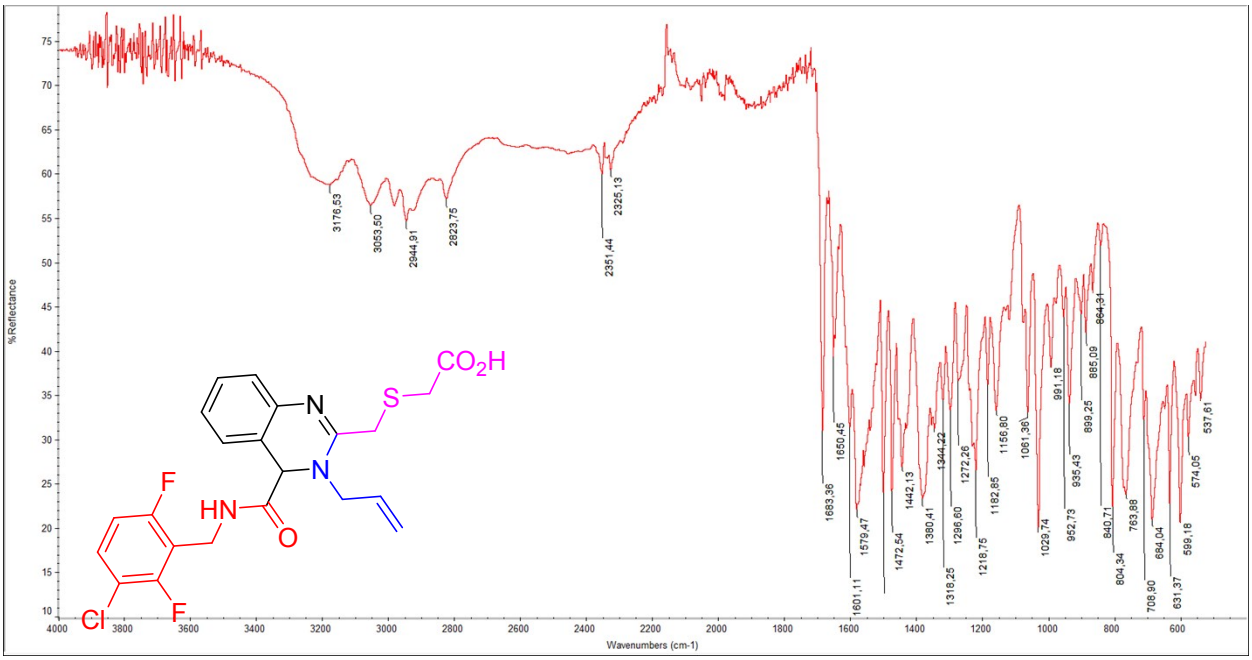




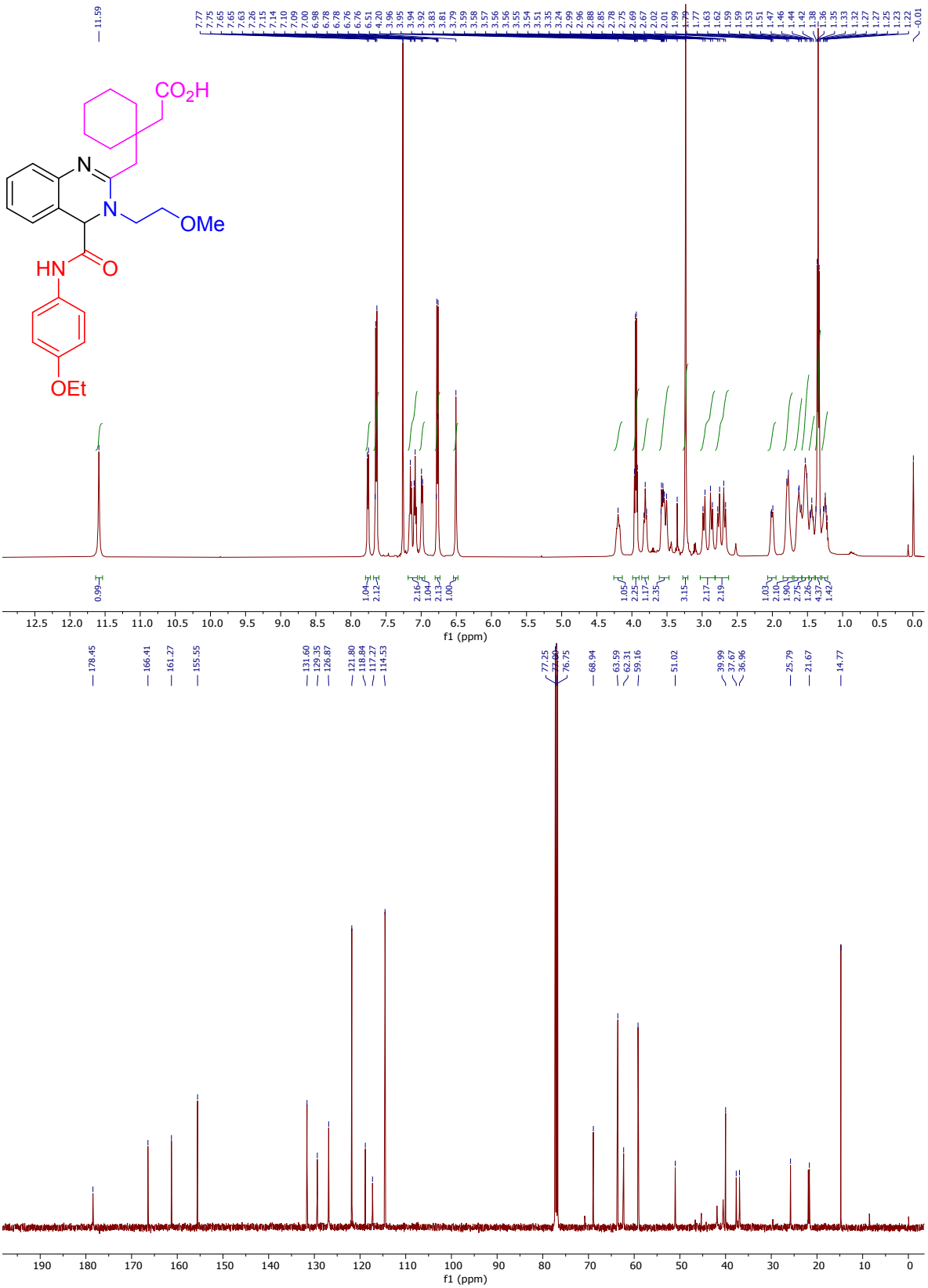

%Reflectance
Wavenumbers (cm-1)
3850,63
3232,52
2915,12
2358,80
2324,83
1681,96
1650,97
1633,57
1564,73
1557,50
1492,69
1453,88
1373,09
1288,25
1241,33
1180,84
1118,12
1074,24
1049,19
1026,69
906,00
813,39
698,54
611,49
577,25
530,86
CO2H
S
N
N
HN
O
OMe

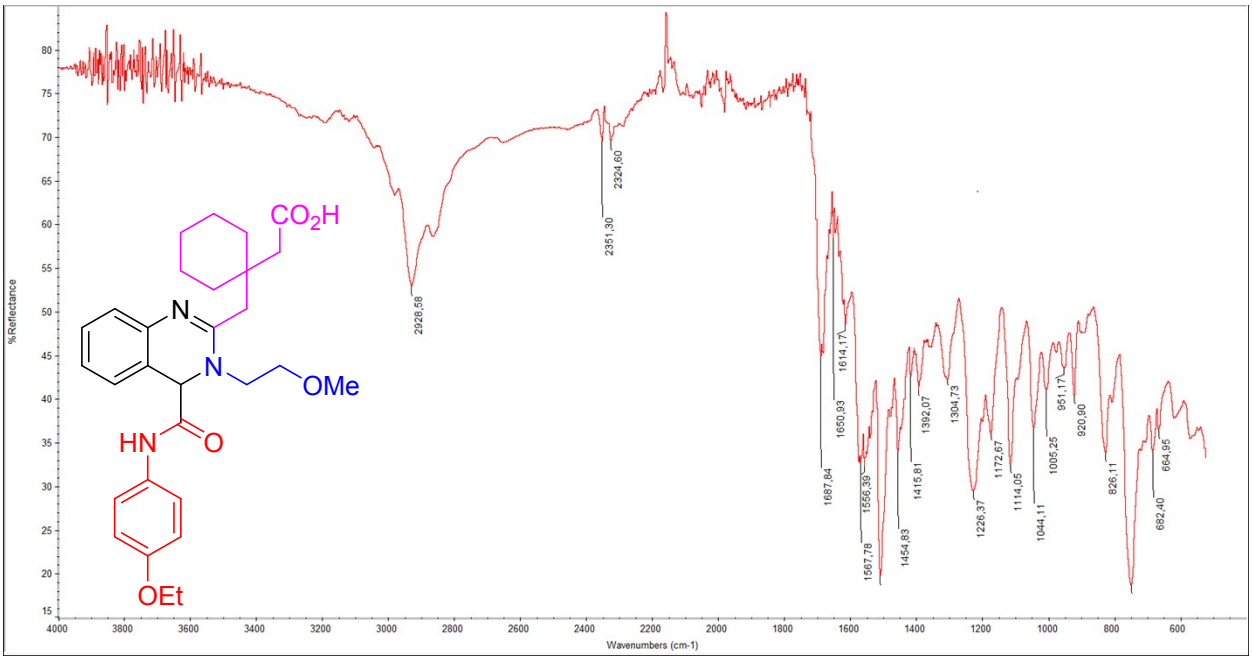
J23: 2-(((3-allyl-4-((3-chloro-2,6-difluorobenzyl)carbamoyl)-3,4-dihydroquinazolin-2-yl)methyl)thio)acetic acid



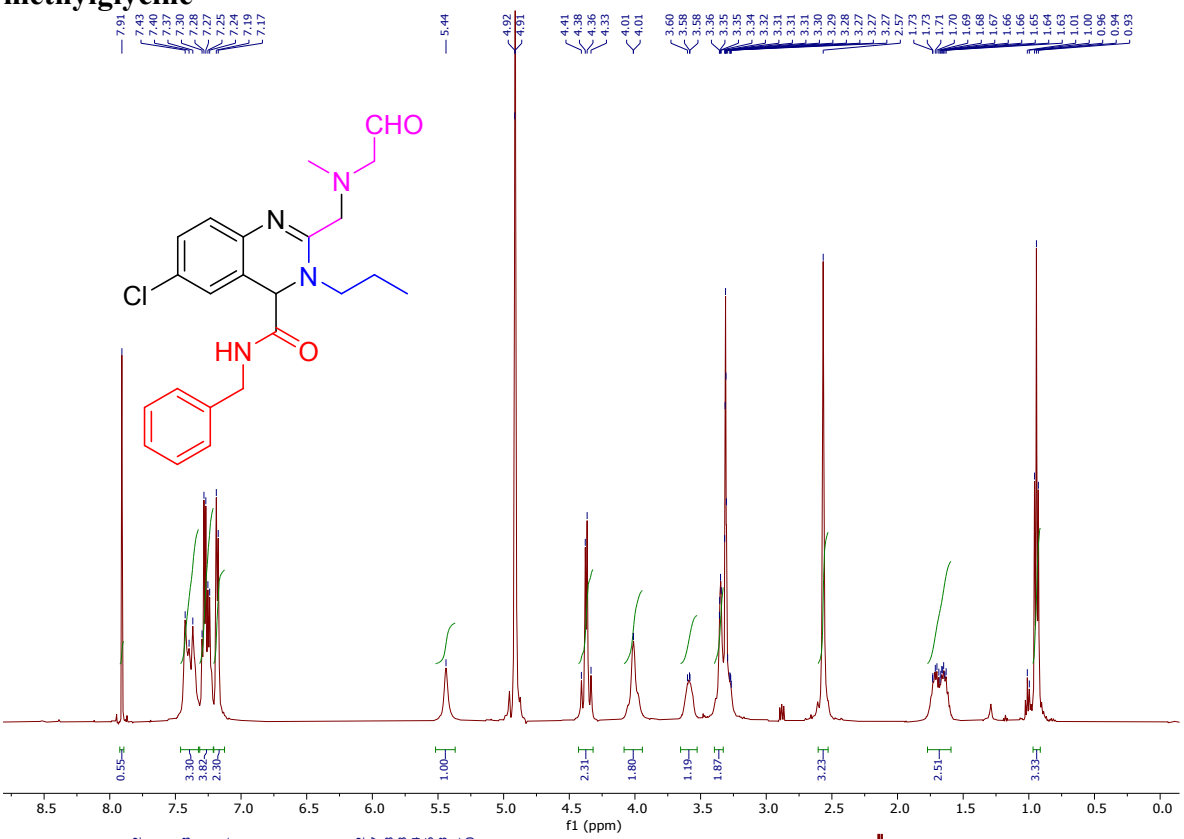

%Reflectance
Wavenumbers (cm-1)
3176,53
3053,50
2944,91
2823,75
2351,44
2325,13
1683,36
1650,45
1601,11
1579,47
1472,54
1442,13
1380,41
1344,22
1318,25
1296,60
1272,26
1218,75
1182,85
1156,80
1061,36
1029,74
991,18
952,73
935,43
899,25
885,09
864,31
840,71
804,34
763,88
708,90
684,04
631,37
598,18
574,05
537,61
CO2H
S
N
N
HN
O
F
F
Cl

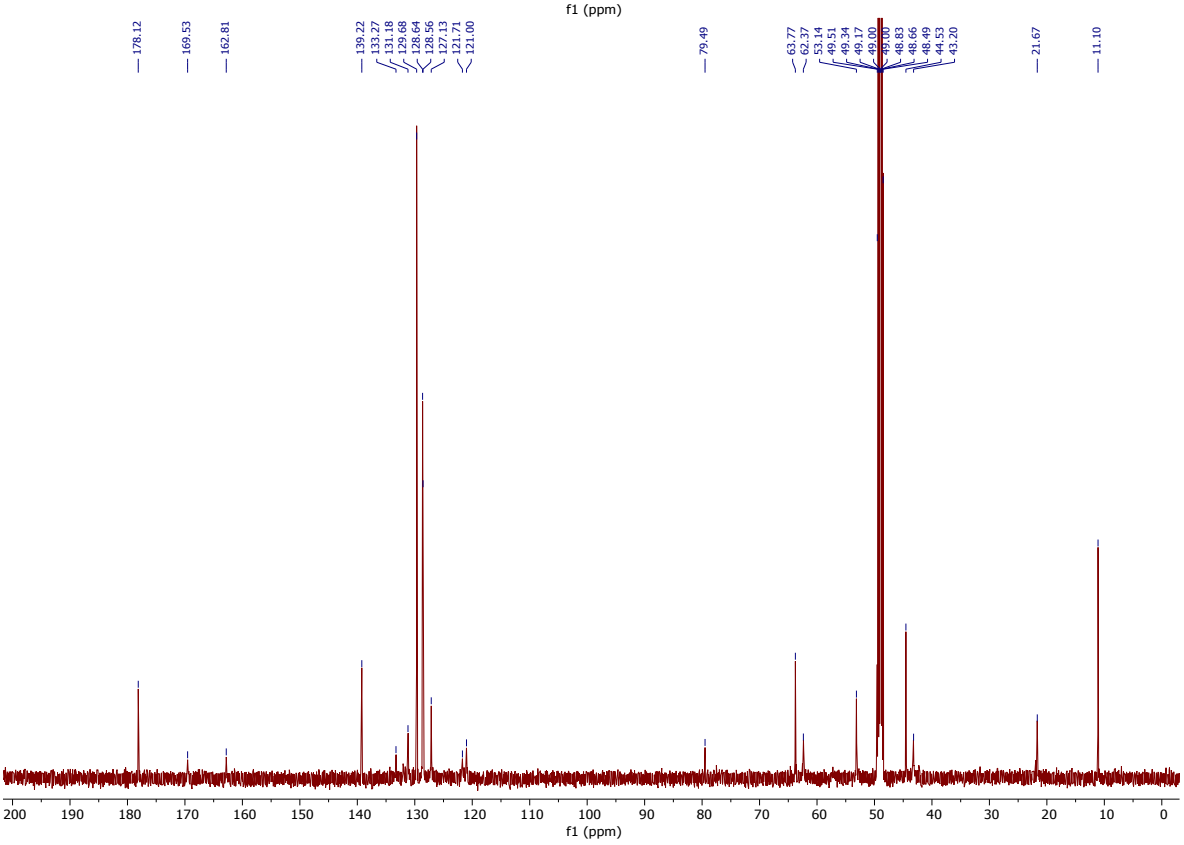
K7:2-(1-((4-((4-ethoxyphenyl)carbamoyl)-3-(2-methoxyethyl)-3,4-dihydroquinazolin-2-yl)methyl)cyclohexyl)acetic acid

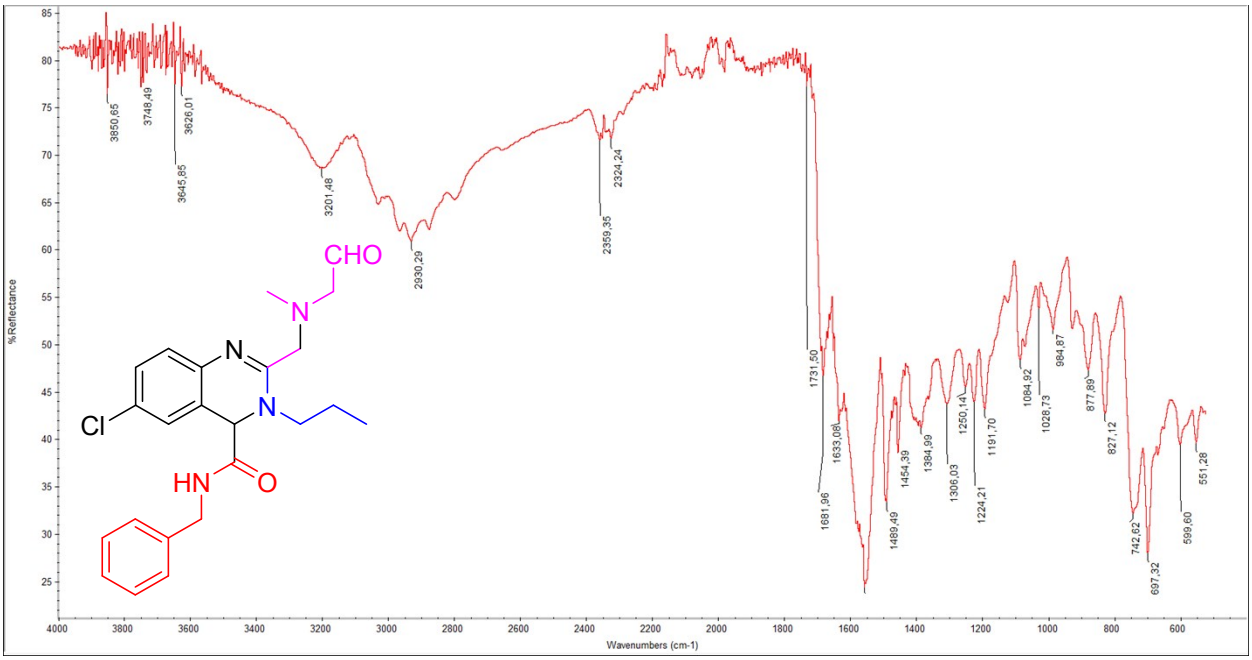



%Reflectance
Wavenumbers (cm-1)
2928,58
2351,30
2324,60
1687,84
1650,93
1614,17
1567,78
1556,39
1454,83
1415,81
1392,07
1304,73
1226,37
1172,67
1114,05
1044,11
1005,25
951,17
920,90
826,11
682,40
664,95
CO2H
N
N
OMe
HN
O
OEt

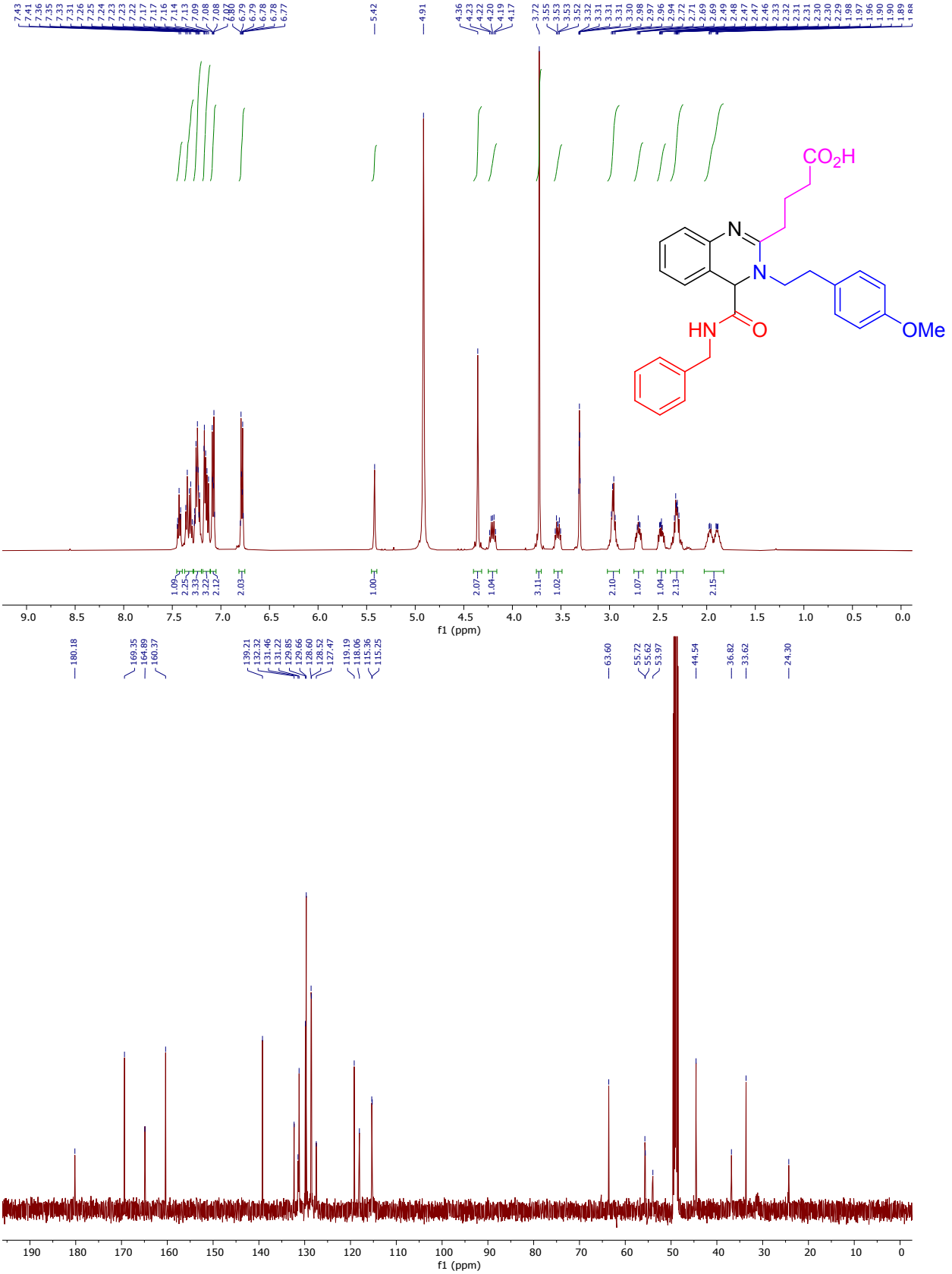
N7:N-((4-(benzylcarbamoyl)-6-chloro-3-propyl-3,4-dihydroquinazolin-2-yl)methyl)-N-methylglycine

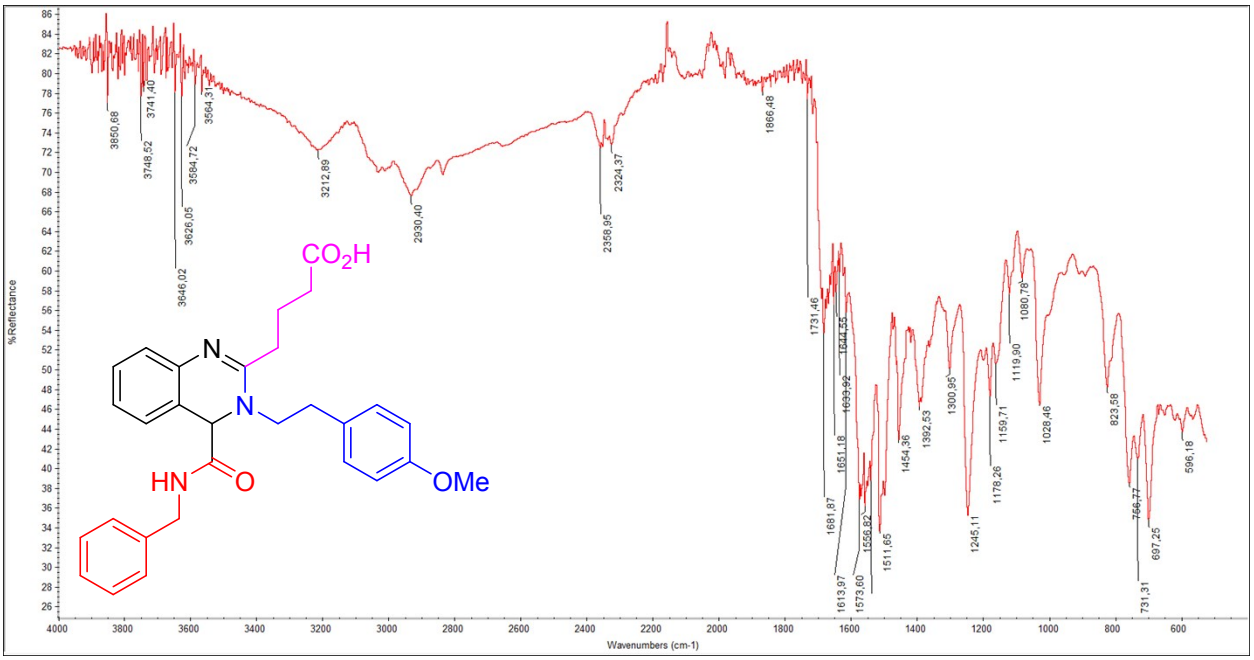





%Reflectance
Wavenumbers (cm-1)
3850,65
3748,49
3626,01
3645,85
3201,48
2930,29
2359,35
2324,24
1731,50
1681,96
1633,08
1489,49
1454,39
1384,99
1306,03
1250,14
1224,21
1191,70
1084,92
1028,73
984,87
877,89
827,12
742,62
697,32
599,60
551,28
CHO
N
N
N
Cl
HN
O

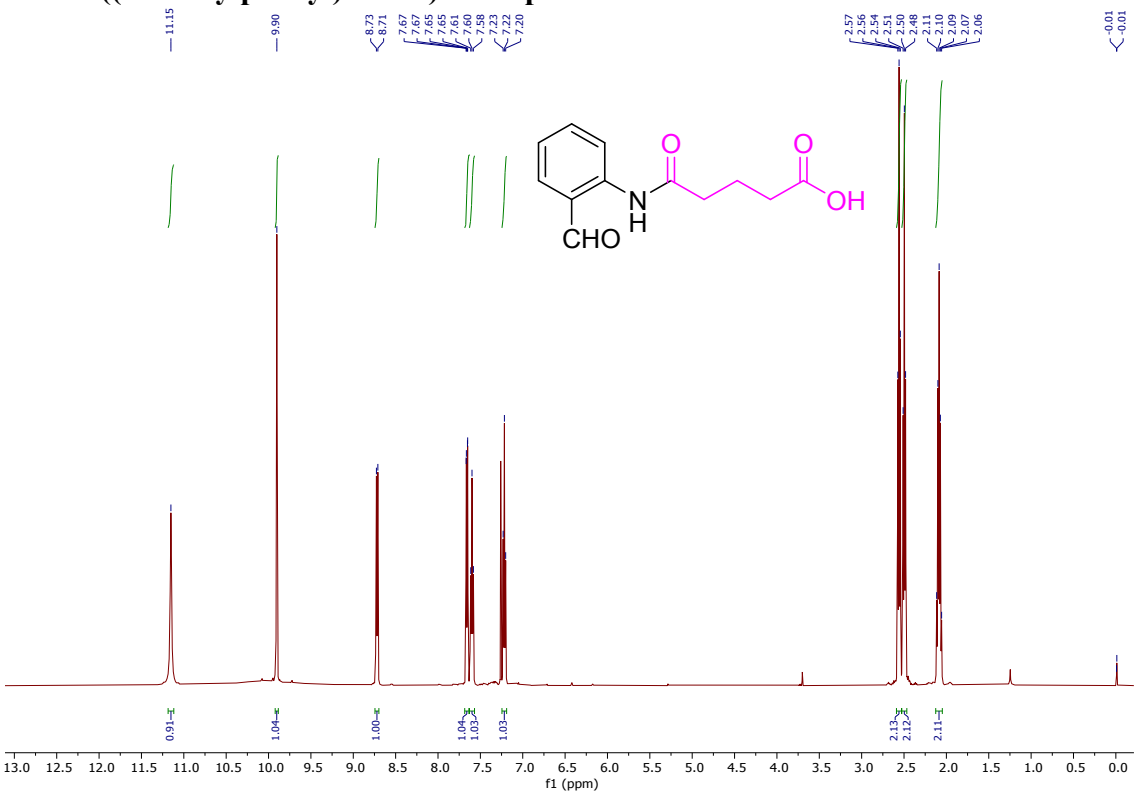
P2:4-(4-(benzylcarbamoyl)-3-(4-methoxyphenethyl)-3,4-dihydroquinazolin-2-yl)butanoic acid

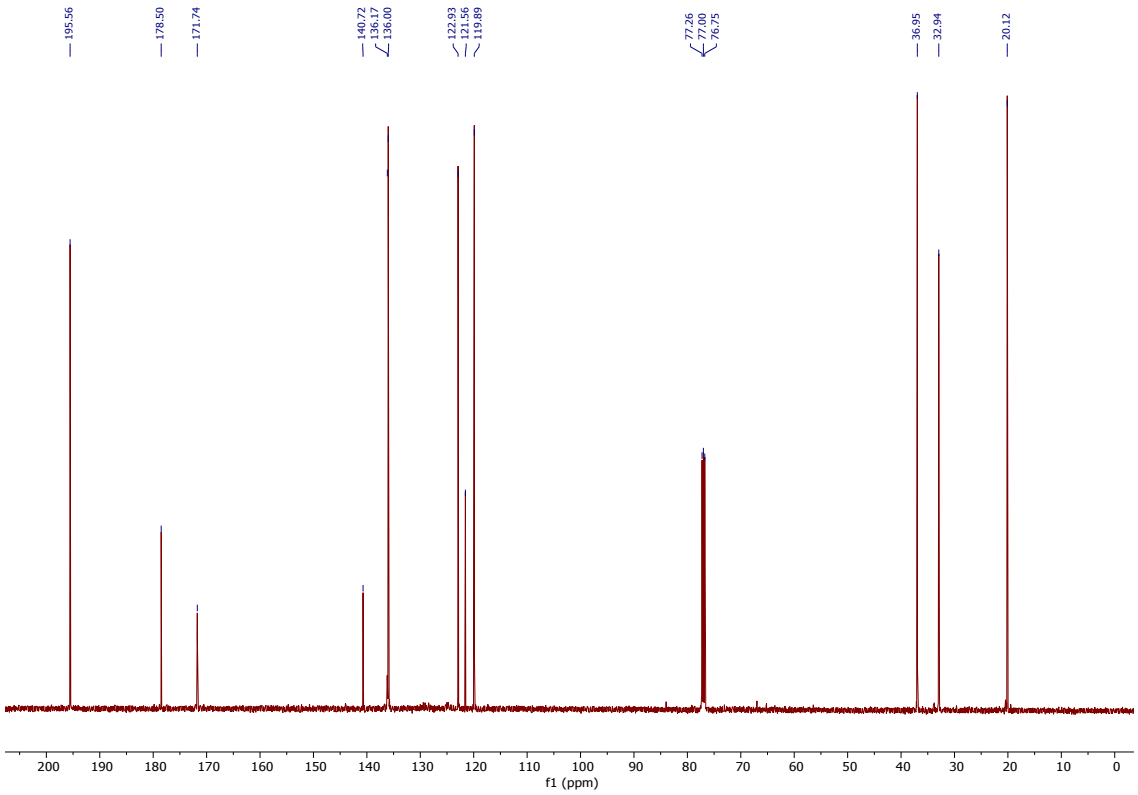



%Reflectance
Wavenumbers (cm-1)
3850,68
3741,40
3748,52
3626,05
3584,72
3564,31
3646,02
3212,89
2930,40
2358,95
2324,37
1866,48
1731,46
1644,55
1699,92
1651,18
1681,87
1613,97
1573,60
1558,82
1511,65
1454,36
1392,53
1300,95
1245,11
1178,26
1159,71
1119,90
1080,78
1028,46
823,58
766,77
731,31
697,25
596,18
CO2H
N
N
HN
O
OMe

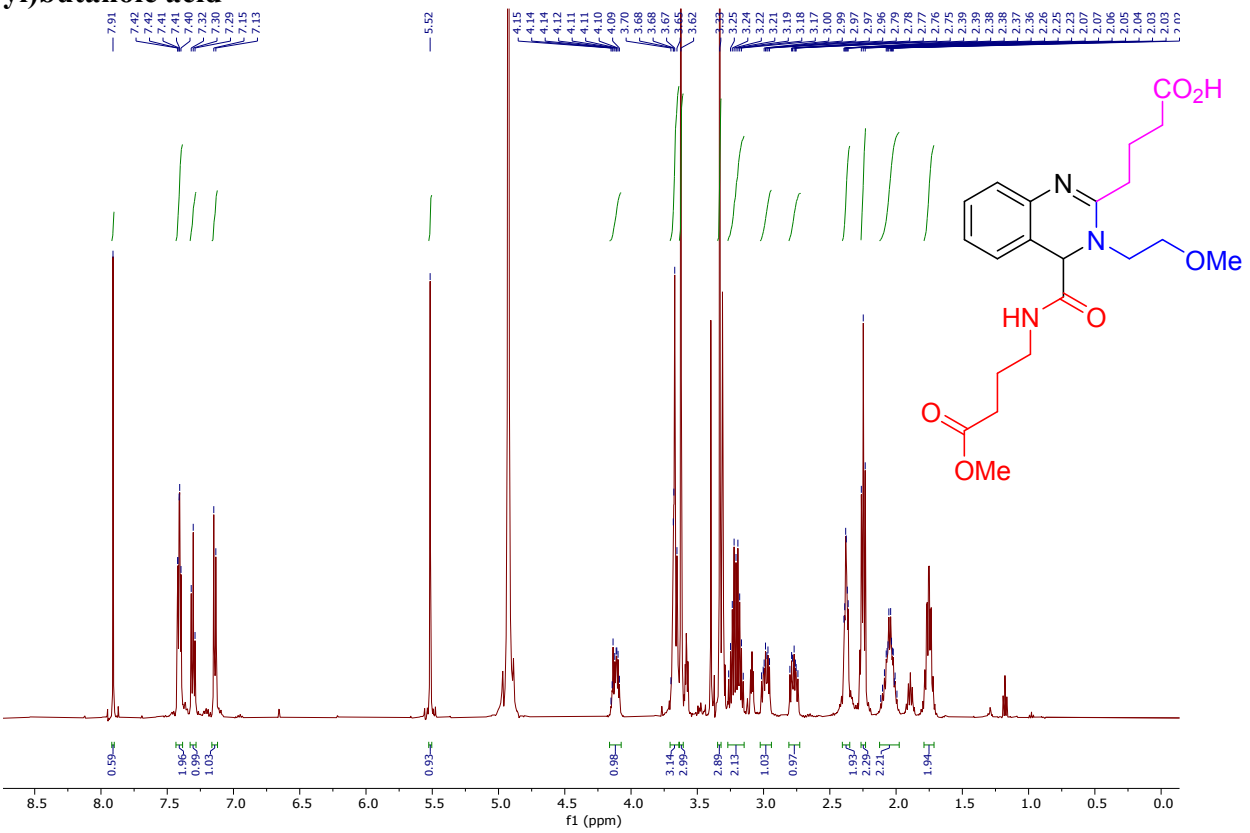
^{1}H and ^{13}C NMR spectra of gram scale reaction

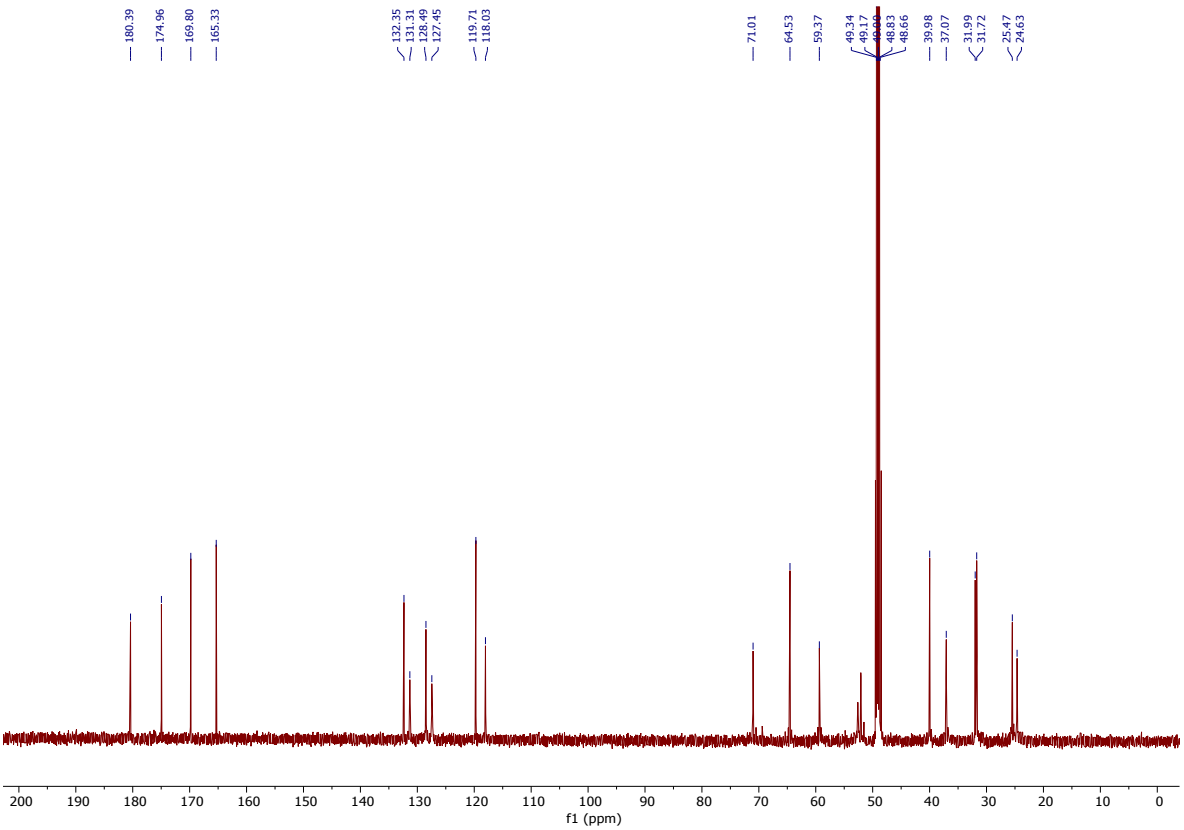
C-2: 5-((2-formylphenyl)amino)-5-oxopentanoic acid





C11: 4-(4-((4-methoxy-4-oxobutyl)carbamoyl)-3-(2-methoxyethyl)-3,4-dihydroquinazolin-2-yl)butanoic acid





7. X-ray experimental analysis

The colourless crystals of **C-8** ($C_{11}H_{11}NO_4S$) are orthorhombic. At 293 K a = 38.544(4), b = 5.1102(9), c = 11.6487(18) Å, V = 2294.4(6) Å^3, M_r = 253.27, Z = 8, space group $Pna2_1$, d_{calc}= 1.466 g/cm^3, mμ($MoK_a\alpha$) = 0.284 mm^{-1}, F(000) = 1056. Intensities of 22235 reflections (5762 independent, R_{int}=0.122) were measured on the«Xcalibur-3» diffractometer (graphite monochromated MoK_α radiation, CCD detector, ω-scanning, $2\Theta_{max}$ = 60°). The structure was solved by direct method using SHELXTL package.[2] Positions of the hydrogen atoms were located from electron density difference maps and refined by "riding" model with $U_{iso} = nU_{eq}$ (n= 1.5 for hydroxyl groups and n=1.2 for other hydrogen atoms) of the carrier atom. Full-matrix least-squares refinement against F^2 in anisotropic approximation for non-hydrogen atoms using 5762 reflections was converged to wR_2 = 0.167 (R_1 = 0.071 for 2728 reflections with F>4σ(F), S = 0.934). The final atomic coordinates, and crystallographic data for molecule **C-8** have been deposited to with the Cambridge Crystallographic Data Centre, 12 Union Road, CB2 1EZ, UK (fax: +44-1223-336033; e-mail: deposit@ccdc.cam.ac.uk) and are available on request quoting the deposition numbers CCDC 1968957).

The colourless crystals of **C-9** ($C_{11}H_{11}NO_5$) are orthorhombic. At 293 K a = 20.486(2), b = 3.9298(4), c = 13.6374(12) Å, V = 1097.91(19) Å^3, M_r = 237.21, Z = 4, space group $Pca2_1$, d_{calc}= 1.435 g/cm^3, mμ($MoK_a\alpha$) = 0.115 mm^{-1}, F(000) = 496. Intensities of 10062 reflections (2644 independent, R_{int}=0.066) were measured on the«Xcalibur-3» diffractometer (graphite monochromated MoK_α radiation, CCD detector, ω-scaning, $2\Theta_{max}$ = 60°). The structure was solved by direct method using SHELXTL package. Positions of the hydrogen atoms were located from electron density difference maps and refined by "riding" model with $U_{iso} = 1.2U_{eq}$ of the carrier atom. Hydrogen atoms of the hydroxyl and amino groups were refined using isotropic approximation. Full-matrix least-squares refinement against F^2 in anisotropic approximation for non-hydrogen atoms using 2644 reflections was converged to wR_2 = 0.137 (R_1 = 0.054 for 1685 reflections with F>4σ(F), S = 1.003). The final atomic coordinates, and crystallographic data for molecule **C-9** have been deposited to with the Cambridge Crystallographic Data Centre, 12 Union Road, CB2 1EZ, UK (fax: +44-1223-336033; e-mail: deposit@ccdc.cam.ac.uk) and are available on request quoting the deposition numbers CCDC 1968958).

The colourless crystals of **C-11** ($C_{11}H_9NO_4$) are monoclinic. At 293 K a = 20.522(5), b = 3.8101(12), c = 26.193(8) Å, β = 102.39(3)°, V = 2000.4(10) Å^3, M_r = 219.19, Z = 8, space group

C2/c, d_{calc}= 1.456 g/cm^3, mμ(MoK_aα) = 0.113 mm^{-1}, F(000) = 912. Intensities of 9140 reflections (2930 independent, R_{int}=0.101) were measured on the«Xcalibur-3» diffractometer (graphite monochromated MoK_α radiation, CCD detector, ω-scaning, $2\Theta_{max}$ = 60°). The structure was solved by direct method using SHELXTL package. Positions of the hydrogen atoms were located from electron density difference maps and refined by "riding" model with U_{iso} = nU_{eq} (n= 1.5 for hydroxyl groups and n=1.2 for other hydrogen atoms) of the carrier atom. Full-matrix least-squares refinement against F^2 in anisotropic approximation for non-hydrogen atoms using 2930 reflections was converged to wR_2 = 0.190 (R_1 = 0.076 for 933 reflections with F>4σ(F), S = 0.858). The final atomic coordinates, and crystallographic data for molecule **C-11** have been deposited to with the Cambridge Crystallographic Data Centre, 12 Union Road, CB2 1EZ, UK (fax: +44-1223-336033; e-mail: deposit@ccdc.cam.ac.uk) and are available on request quoting the deposition numbers CCDC 1968959).

The colourless crystals of **H17 (**$C_{19}H_{23}N_3O_3$) are monoclinic. At 293 K a = 11.864(3), b = 11.162(3), c = 13.889(5) Å, β = 99.31(3)°, V = 1815.1(10) Å^3, M_r = 341.40, Z = 4, space group $P2_1/c$, d_{calc}= 1.249 g/cm^3, mμ(MoK_aα) = 0.086 mm^{-1}, F(000) = 728. Intensities of 12145 reflections (3187 independent, R_{int}=0.177) were measured on the«Xcalibur-3» diffractometer (graphite monochromated MoK_α radiation, CCD detector, ω-scaning, $2\Theta_{max}$ = 50°). The structure was solved by direct method using SHELXTL package. Positions of the hydrogen atoms were located from electron density difference maps and refined by "riding" model with U_{iso} = nU_{eq} (n= 1.5 for methyl groups and n=1.2 for other hydrogen atoms) of the carrier atom. Full-matrix least-squares refinement against F^2 in anisotropic approximation for non-hydrogen atoms using 3136 reflections was converged to wR_2 = 0.129 (R_1 = 0.075 for 1164 reflections with F>4σ(F), S = 0.884). The final atomic coordinates, and crystallographic data for molecule **H17** have been deposited to with the Cambridge Crystallographic Data Centre, 12 Union Road, CB2 1EZ, UK (fax: +44-1223-336033; e-mail: deposit@ccdc.cam.ac.uk) and are available on request quoting the deposition numbers CCDC 1972493).

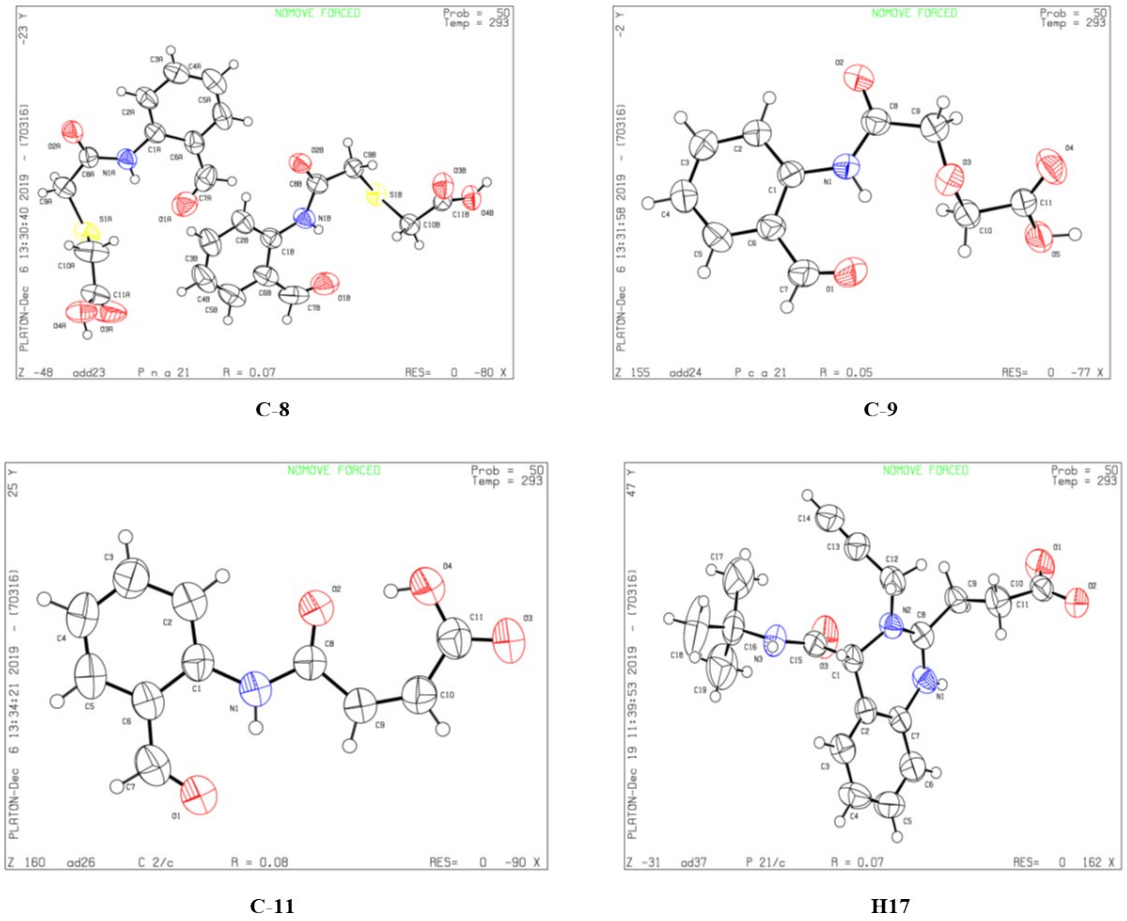


Fig. S7. Molecular structures according to X-ray diffraction study.

References

1. Y. Huang, S. Wolf, M. Bista, L. Meireles, C. Camacho, T. A. Holak and A. Dömling, *Chem. Biol. Drug. Des.*, 2010, **76**, 116-129.
2. G. M. Sheldrick, *Acta Crystallogr., Sect. A*, 2008, **64**, 112-122.