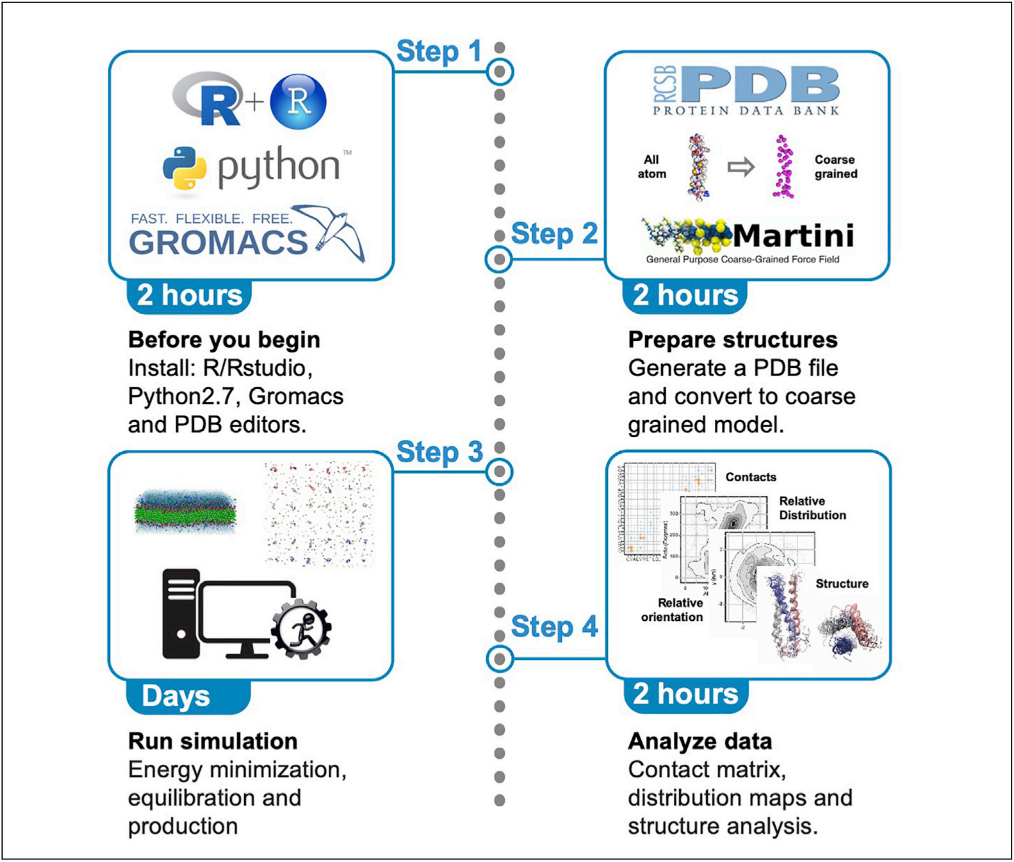
STAR Protocols

CellPress
OPEN ACCESS

Protocol

# Protocol to study the oligomeric organization of single-span transmembrane peptides using molecular dynamics simulations



Herein, you will find detailed information for the preparation of a coarse-grained array of peptides embedded in a lipid membrane. It contains all the steps to set up and run a molecular dynamic simulation using a coarse-grained approach. We provide analytical tools and scripts for generating a residue-level contact matrix between multiple peptides, as well as geometric analysis of arrangements between multiple peptides. This protocol was designed to study the organization of transmembrane peptides in an unbiased manner using computational approaches.

Publisher's note: Undertaking any experimental protocol requires adherence to local institutional guidelines for laboratory safety and ethics.

Mauricio P. Sica, Micaela V. Kortsarz, Angelines A. Morillas, Cristian R. Smulski

cristian.smulski@cab.cnea.gov.ar

Highlights

Protocol to study the oligomeric organization of single-span transmembrane peptides

Step-by-step preparation of a coarse-grained array of peptides in a lipid membrane

Steps for set up and running molecular dynamic simulation using a coarse-grained approach

Tools and scripts for data analysis, contact matrix, and relative orientations

Sica et al., STAR Protocols 3, 101636
September 16, 2022 © 2022 The Author(s).
https://doi.org/10.1016/j.xpro.2022.101636

Protocol

# Protocol to study the oligomeric organization of single-span transmembrane peptides using molecular dynamics simulations

Mauricio P. Sica,[1,3] Micaela V. Kortsarz,[1] Angelines A. Morillas,[2] and Cristian R. Smulski[1,4,*]

[1]Medical Physics Department, Centro Atómico Bariloche, Comisión Nacional de Energía Atómica (CNEA), Consejo Nacional de Investigaciones Científicas y Técnicas (CONICET), Av. E. Bustillo 9500, San Carlos de Bariloche, Río Negro R8402AGP, Argentina

[2]Centro de Investigaciones Energéticas, Medioambientales y Tecnológicas (CIEMAT), Av. Complutense 40, 28040 Madrid, España

[3]Technical contact: mauricio.sica@ib.edu.ar

[4]Lead contact

*Correspondence: cristian.smulski@cab.cnea.gov.ar
https://doi.org/10.1016/j.xpro.2022.101636

## SUMMARY

**Herein, you will find detailed information for the preparation of a coarse-grained array of peptides embedded in a lipid membrane. It contains all the steps to set up and run a molecular dynamic simulation using a coarse-grained approach. We provide analytical tools and scripts for generating a residue-level contact matrix between multiple peptides, as well as geometric analysis of arrangements between multiple peptides. This protocol was designed to study the organization of transmembrane peptides in an unbiased manner using computational approaches.**
**For complete details on the use and execution of this protocol, please refer to Smulski et al. (2022).**

## BEFORE YOU BEGIN

Here we describe a protocol to prepare and perform a molecular dynamics simulation of copies of a peptide embedded in a lipid bilayer. We describe the preparation of a coarse-grained model using the Martini force field. In the Martini coarse-grained model every bead represents a group of atoms (usually 4). There are several types of coarse-grain beads, every type retaining the chemical properties of their corresponding group of atoms, such as polarity, partial charge, possibility of hydrogen-bonding as donor or acceptor and strength of van der Waals interactions. Amino acids are represented with one backbone bead (BB) mapping the backbone heavy atoms N, CA, C and O and additional beads representing the side chains. For example, tryptophan is composed of five beads while glycine and alanine are composed of one BB bead. There is also a solvent bead that maps to four water molecules. Therefore, coarse-graining implies a reduction in the number of particles and covalent bonds which allows simulating complex systems for relatively long periods of time and with minimal loss of structural information (Marrink et al., 2007; Monticelli et al., 2008; Siewert et al., 2019). To reduce the computational time and hardware requirements, this protocol uses a system composed of 9 peptides and a simulation time of 3 microseconds. With the example peptide provided and this setup you will obtain results comparable to our original paper, where we simulated 36 peptides for more than 8 microseconds (Sica and Smulski 2021; Smulski et al., 2022). Consider that with other peptides you may need a larger setup and longer simulation times as in the original paper to broadly explore the interaction landscape.

### Installing software

Timing: 2 h

To perform this protocol, you will need:

1. A computer with Linux OS (operative system) and internet access.
2. The following software installed in your computer:
   a. GROMACS, version 2016 or superior (Abraham et al., 2015).
   b. R, version 3 or superior (https://www.R-project.org/).
   c. DSSP, version 2.0.4 or superior (Touw et al., 2015; Kabsch and Sander 1983).
   d. Python 2.7.
   e. martinize.py (de Jong et al., 2013): version 2,6. It is a Python script obtained from http://www.cgmartini.nl/.
   f. insane.py (Wassenaar et al., 2015). It is a Python script obtained from http://www.cgmartini.nl/.
3. We also recommend the installation of the following software that is not essential to perform this protocol:
   a. RStudio, version 3 or superior. It is helpful to run and debug the R scripts (http://www.rstudio.com/).
   b. VMD (Humphrey et al., 1996) or pymol (http://www.pymol.org/pymol) for the visualization.
   c. Git, a version control system that helps to download the whole repository.

GROMACS, R, Python 2.7, DSSP, pymol and git are available in the repositories of Linux distribution Debian and Ubuntu. To install them, open a terminal and type the next commands:

```
> sudo apt install gromacs

> sudo apt install r-base

> sudo apt install python2.7

> sudo apt install dssp

> sudo apt install pymol
```

## KEY RESOURCES TABLE

| REAGENT or RESOURCE | SOURCE | IDENTIFIER |
|---|---|---|
| Software and algorithms | | |
| GROMACS | http://www.gromacs.org/ | RRID:SCR_014565 |
| PyMOL | http://www.pymol.org/ | RRID:SCR_000305 |
| DSSP | https://swift.cmbi.umcn.nl/gv/dssp/ | RRID:SCR_002725 |
| Python 2.7 | https://www.python.org/ | RRID:SCR_008394 |
| R | https://www.r-project.org/ | RRID:SCR_001905 |
| RStudio | http://www.rstudio.com/ | RRID:SCR_000432 |
| ggplot2 R package | https://www.rdocumentation.org/packages/ggplot2/versions/3.3.6 | RRID:SCR_014601 |
| reshape R package | https://cran.r-project.org/web/packages/reshape/index.html | RRID:SCR_018983 |
| Martini 2 | http://cgmartini.nl/index.php | N/A |
| martinize.py 2.6 | http://cgmartini.nl/index.php/tools2/proteins-and-bilayers/204-martinize | RRID:SCR_022318 |
| insane.py | http://www.cgmartini.nl/index.php/downloads/tools/239-insane | RRID:SCR_022317 |
| GitLab | https://about.gitlab.com | RRID:SCR_013983 |
| Other | | |
| Tools and scripts related to the protocol | This protocol | https://gitlab.com/BioComp/taci_cg_simulations |

## MATERIALS AND EQUIPMENT

This protocol was designed to be performed on a desktop computer with a Linux OS (operative system) installed. You can also have a computer with another OS and use Linux OS in a virtual machine. The computer must also have Internet access.

To perform this example, the computer needs to have at least 8GB RAM and a CPU similar or superior to Intel Corei5. If you intend to use this protocol to run longer simulations with a higher number of peptides, as in our original paper, consider using a cluster where you can use at least 10-14 cores per simulation.

## STEP-BY-STEP METHOD DETAILS

### Molecular model preparation

Timing: 2 h

Starting from the helical structure (all-atom) of the transmembrane segment of human TACI (Transmembrane activator and calcium modulator and cyclophilin ligand interactor, TNFRSF13B), you will create the corresponding coarse-grained version. Then you will prepare an array of 9 peptides equidistantly distributed and randomly oriented around the z-axis. This array will be embedded in a lipid bilayer composed of dioleyl palmitoyl choline (DOPC) and dilauryl palmitoyl choline (DLPC) (7:3) equally distributed on both sides of the membrane. Finally, the system will include water particles and ions. This system will be used in the following steps for the molecular dynamic simulation.

All the scripts and accessory files necessary for this protocol are available from our repository in GitLab.

1. Download the whole repository from this link (https://gitlab.com/BioComp/taci_cg_simulations) as a compressed file or, if you use git, clone the repository with the next command:

```
> git clone https://gitlab.com/BioComp/taci_cg_simulations
```

2. Once the repository is decompressed as a folder in your computer, open a bash terminal and move there with the following command, where PATH is the location in your computer where you saved the content of the repository:

```
> cd PATH/taci_cg_simulations
```

3. Create a working directory named as you prefer ("`PROT`" in this example):

```
> mkdir PROT
```

4. From the folder "`example_files`" copy the file "`peptide.pdb`" into the folder PROT using this command:

```
> cp example_files/peptide.pdb PROT/
```

*Note:* In this example, ``peptide.pdb'' corresponds to the helical structure of the human TACI transmembrane domain (residues 156 to 191) structure from AlphaFold (AF-O14836-F1).

5. Also copy the whole folder ``martiniff''. This folder contains the version of the Martini force field and the ``lipids.itp'' file used in our original work.

```
> cp -r martiniff/ PROT/
```

6. Move to the new working directory:

```
> cd PROT
```

   a. Once you are in this folder, download a copy of the files ``martinize.py'' (http://cgmartini.nl/index.php/tools2/proteins-and-bilayers/204-martinize) and ``insane.py'' (http://www.cgmartini.nl/index.php/downloads/tools/239-insane) inside this folder. For using these scripts, you have to grant them execution permission with the next commands:

```
> chmod +x martinize.py

> chmod +x insane.py
```

*Note:* The next instructions assume you are using ``martinize.py'' version 2.6 and the current version of ``insane.py'', the same versions we used in our work. Using other versions, especially versions 3+, may have some changes in the syntax and requirements. Check the manuals in the Martini force field webpage.

   b. Be sure that the DSSP program (https://swift.cmbi.umcn.nl/gv/dssp/) is correctly installed on your computer. DSSP is part of the Debian and Ubuntu Linux distributions. Once installed, you can copy the path of the executable file from the output of the next command:

```
> which dssp
```

Copy this path and use it as the variable for the flag ``-dssp'' in the next step. Usually the path is ``/usr/bin/dssp''.

*Note:* If the previous command does not provide a path, try "which mkdssp" instead. If the command provides a valid path, use it for the option "-dssp" in the next step.

7. Convert an all-atom model of the peptide into a coarse-grain MARTINI model with the next command:

```
> martinize.py -f peptide.pdb \

     -o peptide-CG.top \

     -x peptide-CG.pdb \
     -ff elnedyn22 \
     -name Protein \
     -p None \
     -elastic -cys auto -eu 1.0 \
     -dssp /usr/bin/dssp \
     -nt \
     -sep
```

As a result, you will obtain a topology file named ``peptide-CG.top'' and a structure file named ``peptide-CG.pdb''. Troubleshooting 1.

*Note:* ``martinize.py'' version 2,6 runs in Python 2.7. If you have several Python versions in your system, you may have to add ``Python2.7'' before ``martinize.py''. If you use conda environments with another Python version, deactivate conda.

You can check the ``martinize.py –help'' for options. With the option ``-ff'' you can use any of the force fields available in the ``martinize.py'' script: martini21, martini21p, martini22, martini22p, elnedyn, elnedyn22 and elnedyn22p. In this example we use elnedyn22. You can perform this step also with the new version of martinize3, which is a Python package.

*Optional:* You can open the ``peptide.pdb'' and ``peptide-CG.pdb'' files with pymol to visualize the input structure and the generated coarse-grained model. To do this, first open pymol and then type the following lines in the pymol console.

```
> load PATH/peptide.pdb
> load PATH/peptide-CG.pdb
> show spheres
> bg_color white
```

You must replace ``PATH'' with the path where ``peptide.pdb'' and ``peptide-CG.pdb'' are located in your computer. You will see structures like the ones shown in Figure 1.

8. Using the coarse-grained peptide structure ``peptide-CG.pdb'' previously generated, build a box where a single helix is aligned to the *z*-axis using the following command:

```
> gmx editconf -f peptide-CG.pdb -c -princ -o aligned_peptide.gro
```

   a. Choose ``System (option 0)'' as the group for determining the orientation.

*Note:* All GROMACS tools are preceded by the “gmx” command. Depending on your installation, this prefix can vary. A usual variant is “gmx_mpi”.

9. Rotate the peptide 90 degrees around the y-axis (or x-axis, if necessary):

```
> gmx editconf -f aligned_peptide.gro -c -rotate 0 90 0 -o z_aligned_peptide.gro
```

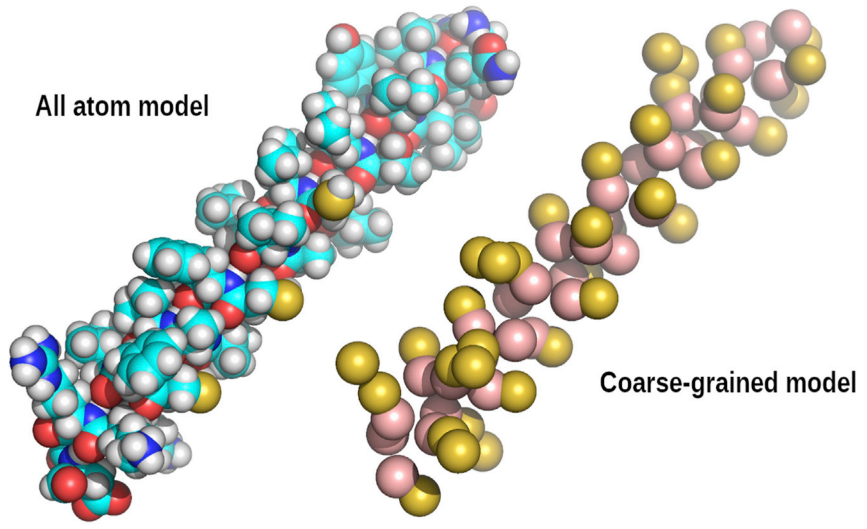


**Figure 1. All-atom and coarse-grained peptide models**
Pymol visualization of ``peptide.pdb'' (all-atom model) and ``peptide-CG.pdb'' (coarse-grained model) files. Colors in the all-atom model are the default scheme from Pymol : C, H, O, N, O, S, in cyan, white, red, blue and yellow, respectively. In the coarse-grained model, BB beads are colored in pink and side chain beads in yellow.

a. The location of the N-terminus of the peptide (towards the upper or the lower membrane layer) can be controlled by rotating plus or minus 90 degrees. You can visualize the orientation of the peptide in VMD by using the reference *XYZ* axes on the bottom left corner of the model display window as follows:

```
> vmd -m z_aligned_peptide.gro
```

b. In the menu ``Graphics/Representations'' you can create a representation of the BB atoms (type ``name BB'' in the Selected Atoms box) and choose ``VDW'' as the Drawing Methods to observe the BB particles as van der Waals spheres.

10. Create an array of 3 x 3 peptides where the helices are located equidistantly and randomly oriented around their *z*-axis using the following command:

```
> gmx genconf -f z_aligned_peptide -nbox 3 3 1 -dist 4 4 0 -o peptide_array.gro -rot -maxrot
0 0 180
```

*Note:* The peptide array is specified in the option ``-nbox 3 3 1''. The distance between peptides in the *xy*-plane is controlled by the option ``-dist 4 4 0''. The options ``-rot -maxrot 0 0 180'' ensure that every peptide is rotated randomly around the *z*-axis ±180 degrees. In the simulation box, the membrane plane is parallel to the XY plane, so the principal axis of the transmembrane peptides must be oriented parallel to the Z-axis. The starting configuration is set up to avoid initial peptide-peptide interactions. Thus, peptides should be completely surrounded by lipid molecules and distributed on the XY plane separated from each other by a distance greater than the cutoffs of the vdW and coulombic interactions (specified in the file production_310K.mdp). In addition, to avoid a preferential orientation of the first contacts as the peptides approach each other, the helices must be randomly rotated around the Z-axis. These features allow an unbiased exploration of the environment.

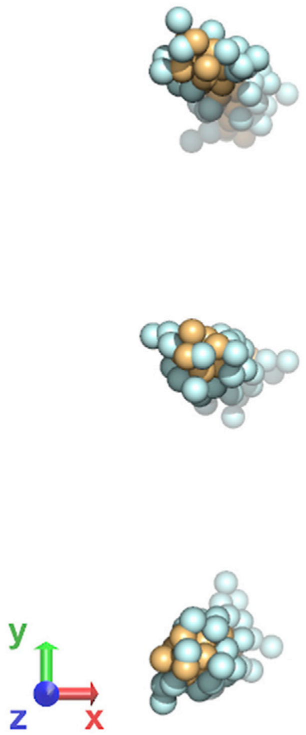


Figure 2. Peptide array
Pymol visualization of "peptide_array.gro" file. BB beads are in orange, side chain beads in light blue.

⚠ CRITICAL: In the previous steps you have created several files with the ``.gro'' extension. Although some of them are temporary files or intermediate steps, do not erase them. They can be used in the analysis steps.

*Optional:* As explained before, if you visualize the ``peptide_array.gro'' file with pymol, the structure will be like the one shown in Figure 2.

11. Use the ``insane.py'' script to create a membrane where this array will be embedded.

```
> insane.py -f peptide_array.gro \
     -o peptide_array_mem.gro \
     -p topol-insane.top \
     -salt 0.15 \
     -ring \
     -x 12 -y 12 -z 10.5 \
     -sol W -rand 0.095 \
     -l DLPC:3 -l DOPC:7 \
     -u DLPC:3 -u DOPC:7
```

*Note:* Options ``-x -y -z'' control the dimensions (nm) of the periodic box. The compositions of the lower and upper layers of the membrane are selected with the options ``-l'' and

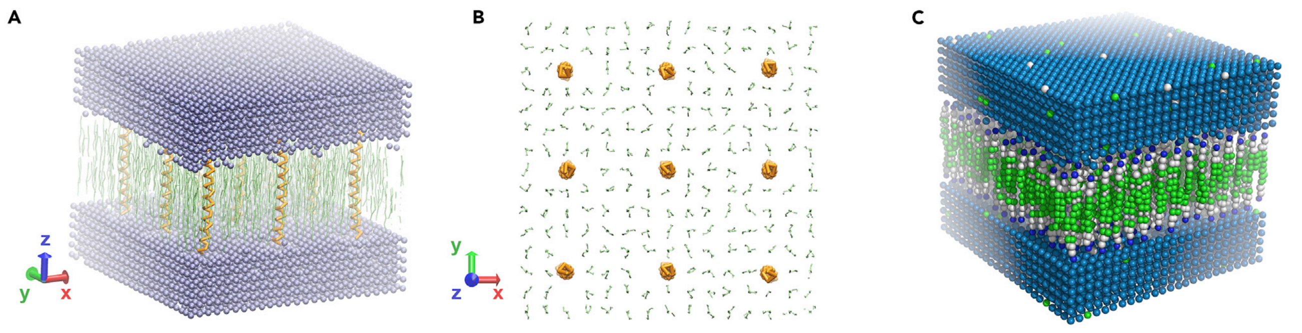


**Figure 3. Simulation box**
Side (A) and top (B) views of the simulation box ("peptide_array_mem.gro" file) in the starting configuration. Waters are shown as violet beads, lipids as green sticks and backbone beads bonded by orange sticks (side chains and ion beads are not shown for simplicity). XYZ axes are shown in every panel. Water beads are not shown in panel (B). The image was generated using VMD and the script leading to this representation can be found in the repository.
(C) Pymol (default) visualization of ``peptide_array_mem.gro'' file. Color scheme used by Pymol may vary with software version and may not correctly represent the different types of beads.

``-u''. We chose DOPC and DLPC (7:3) to mimic the composition of the experimental membranes used to determine the NMR structure of other transmembrane TNFR domains. For alternative membrane compositions, check that the topologies of the lipids you select are included in the ``lipids.itp'' file.

⚠ CRITICAL: To calculate the dimensions (*L*) of the periodic box in the *X* and *Y* axes, consider that the distance from the peptides in the borders of the array to the nearest peptide in the periodic image has to be no less than the separation you specified in the previous command. So, "*L*" needs to be equal or greater than "*N* x *d*", where "*N*" is the number of copies of the peptides in a row or column of the array and "*d*" is the distance you specified in the previous command. The dimension in the *Z* axis is enough to create a layer of water beads that separate the membrane from its periodic image. Then, if you want to create a different array, you have to edit the *XY* values. For example, if you choose ``-nbox 4 4 1 -dist 4 4 0'' in the previous command, here you may use ``-x 16 -y 16 -z 10.5''.

*Optional:* After the previous steps, the initial configuration of the system is composed as shown in the Figures 3A and 3B. The script leading to this representation can be found in the repository. If you visualize the ``peptide_array_mem.gro'' file with pymol, you will see a structure like the one shown in the Figure 3C.

12. Edit the topology file ``.top'' generated by ``insane.py''. This file may not be consistent with the environment variables and the next steps could be aborted by errors. Open ``topol-insane.top'' with a plain text editor (for example ``gedit'' in Linux) and edit as follows.
    a. Open the file in gedit

```
> gedit topol-insane.top
```

    b. Remove the ``#include'' lines (usually the first one or two lines).
    c. Copy the following lines at the beginning of the file:

```
#include "martiniff/martini_v2.1.itp"

#include "martiniff/lipids.itp"

#include "martiniff/martini_v2.0_ions.itp"
```

```
#include "Protein_A.itp"

#ifdef POSRES

#include "posre.itp"

#endif
```

*Note:* Do not exclude the numeral symbol when copying the previous lines. They are not comment symbols. In the GROMACS syntax the comment symbol is ";").

d. You also need to edit two variables in this file:
   i. The internal name of the peptide. This name is defined under the section ``[molecule-type]'' in the file "Protein_A.itp". Copy that name.
   ii. The number of copies of the peptide. In this example we used 9 peptides in the system.
e. Under the section ``[ molecules ]'' in ``topol-insane.top'' you will find a first uncommented line with the name and number of proteins. Be sure that the protein name corresponds to the one you copied in ``Protein_A.itp'' (step 12.d.i) and modify the number of peptides to 9 as follows:
   i. `Protein 1` → `Protein_A 9`

13. Generate an index file with the atom number of the backbone (BB) particles.
   a. In the box below you have an interacting version of the command, where the options are already defined between the flags ``<<OPT'' and ``OPT''.
   b. Copy the four lines and press ``ENTER'' to run the script.
   c. If you only type the command (i.e., first line except ``<<OPT'') the script will ask you to type your options. Then, you have to type the lines 2 and 3 only. You can use this option to have a more GROMACS-like experience.

```
> gmx make_ndx -f z_aligned_peptide.gro -o index_BB.ndx <<OPT

aBB

q

OPT
```

*Note:* GROMACS has an interactive tool to make indices: ``make_ndx''. In the help menu, you can check the examples to learn the syntax and choose your groups of atoms.

14. Generate a position restriction file that will be useful for the preparation steps before the simulation.
   a. Run the command ``genrestr'' to generate a position restriction file using the index file generated in the previous step.
   b. The command asks you to choose an option, select the option corresponding to the group you created (the name of the groups can be ``BB''; check the output of the previous command ``make_ndx'').

```
> gmx genrestr -f peptide_array_mem.gro -n index_BB.ndx
```

## Molecular dynamics simulation

Timing: days or weeks

In this section you will be guided to use the previously generated peptide array in a lipid bilayer for the molecular dynamic simulation. To do so, it is necessary to perform an initial minimization of the system, followed by a position restrained equilibration step and a free equilibration step. Finally, the production step will generate the simulation of the array for 3 microseconds.

15. Energy minimization:
    a. Create nested folders for every step to avoid the accumulation of many files in a single folder. In the working directory ("`PROT''`), create a directory for the minimization and then move there.

```
> mkdir minimization

> cd minimization
```

    b. GROMACS requires creating a ``` ``.tpr'' ``` as input file to run the energy minimization. This is done with the ``` ``grompp'' ``` preprocessor tool that reads the topology and the coordinates, and writes the atomic description of the system.
        i. Copy the file ``` ``minimization.mdp'' ``` from the ``` ``mpds'' ``` folder from our repository into this directory:

```
> cp ../../mdps/minimization.mdp ./
```

        ii. Run the following command:

```
> gmx grompp -f ./minimization.mdp \
      -c ../peptide_array_mem.gro \
      -r ../peptide_array_mem.gro \
      -p ../topol-insane.top
```

*Note:* Most probably, ``` ``grompp'' ``` will halt before creating the output file because of a warning message. Check the importance of the warning. Solve the problem if necessary or surpass this message adding the option ``` ``-maxwarn 1'' ``` to ``` ``grompp'' ```.

    c. Once the ``` ``topol.tpr'' ``` file has been created, run the minimization with the following command.

```
> gmx mdrun -s topol.tpr
```

*Note:* This step can take some minutes to finish.

16. Equilibrations:
    a. Perform a position-restrained equilibration (POSRE) using the output configuration from the minimization.

i. Create a new directory inside ``minimization'' and then move there. Then, copy the file "equilibration_POSRE.mdp" from the "mpds" folder:

```
> mkdir POSRE
> cd POSRE
> cp ../../../mdps/equilibration_POSRE.mdp ./
```

ii. Run the following commands:

```
> gmx grompp -f ./equilibration_POSRE.mdp \
      -c ../confout.gro \
      -r ../confout.gro \
      -p ../../topol-insane.top
> gmx mdrun -s topol.tpr
```

*Note:* This step can take some minutes to finish.

b. Perform a free equilibration step. Create the corresponding directory and move there and copy the file "equilibration_FREE.mdp" from the "mdps" folder:

```
> mkdir equil
> cd equil
> cp ../../../../mdps/equilibration_FREE.mdp ./
```

i. Run the following commands:

```
> gmx grompp -f ./equilibration_FREE.mdp \
      -c ../confout.gro \
      -r ../confout.gro \
      -p ../../../topol-insane.top -maxwarn 4
> gmx mdrun -s topol.tpr
```

*Note:* This step can take some minutes to finish.

17. Production:
    a. Recommended: before running the production simulation, check that the system is properly equilibrated. Please refer to the GROMACS user guide (https://manual.gromacs.org/documentation/current/user-guide/index.html) for good practices and tips in this regard.

    If equilibration steps were successful, we are ready to initiate the production simulation.
    b. Create a new directory, move there, and copy the ``production_310K.mdp'' file from the "mpds" directory. Then, run the ``grompp'' and ``mdrun'' tools as follows.

Depending on your CPU, this step can take from 1 to 7 days.

The procedure is similar to the previous ones and the necessary commands are in the next box.

```
> mkdir production
> cd production
> cp ../../../../../mdps/production_310K.mdp ./
# Download 'production_310K.mdp' into this directory
> gmx grompp -f ./production_310K.mdp \
        -c ../confout.gro \
        -r ../confout.gro \
        -p ../../../../topol-insane.top
> gmx mdrun -s topol.tpr
```

*Optional:* You can parallelize the simulation by adding options to ``mdrun''. These options strongly depend on the structure of your hardware. We recommend checking the GROMACS user guide (https://manual.GROMACS.org/current/user-guide/index.html). If your computer has 4 cores and OpenMPI installed, a usual example could be:

```
> gmx mdrun -ntomp 4 -s topol.tpr
```

c. To observe the simulation use the following command:

```
> gmx view -f traj_comp.xtc -s topol.tpr
```

*Optional:* It is possible to visualize the simulation in VMD. Use the ``confout.gro'' file from the equilibration so you do not need to wait for the simulation to finish. For this use the following command:

```
> vmd -f ../confout.gro traj_comp.xtc
```

In the menu ``Graphics/Representations'' you can create a representation of the BB atoms (type ``name BB'' in the selected atoms box) and choose ``VDW'' as the Drawing Methods to observe the BB particles as van der Waals spheres and the rest of the system as points. One initial step may look like Figure 4.

## Contact map analysis

Timing: 1 h

Here we describe the procedure to calculate a matrix of contacts between peptides at the residue level. For every time step analyzed, a contact is considered to be formed when the distance between two backbone beads (BB) of different peptides is equal to or lesser than the distance cutoff.

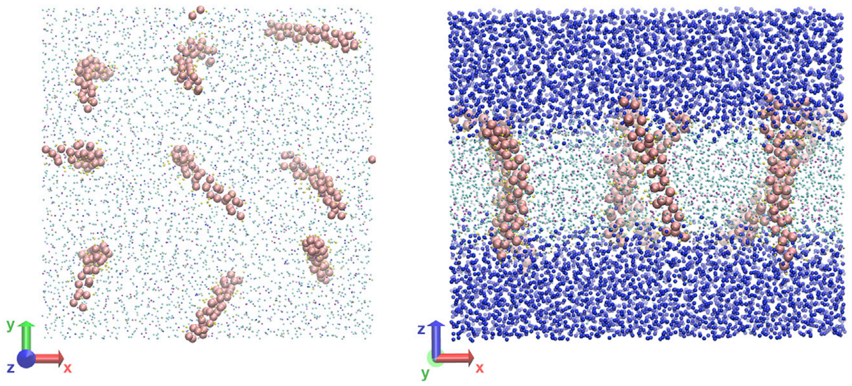


**Figure 4. Trajectory snap shot**
Top (left) and side (right) views of an early snapshot from the trajectory file "traj_comp.xtc" produced with VMD (follow instructions in the repository to reproduce this representation). BB particles are represented as pink spheres. Lipid beads are represented as small spheres of different colors: phosphates in violet, non-polar beads of lipids in green. Water beads are not shown in the top view.

Chemical properties are not explicitly considered to count a contact. This procedure counts how many times along the simulation a BB(i) of every peptide contacts a BB(j) of every other peptide (different from itself). Here, "i" and "j" refer to the number of the residue in the peptide sequence.

In detail, for each residue (i) of every peptide (H) it is computed along the simulation time (T) the number of contacts against all the other residues (j) in the remaining peptides (K={1…N}, K≠H). For a particular time step (t), a contact (c) was defined between residues "i" and "j" from two different peptides when their BB are located at *XYZ*-distance equal to or less than an arbitrary cut-off, as follows:

$$c_{ij}^{HK}(t) = \begin{cases} 1, & if\ \|\mathbf{r}_i^K(t) - \mathbf{r}_j^H(t)\| \le d_{\text{cutoff}} \\ 0, & if\ \|\mathbf{r}_i^K(t) - \mathbf{r}_j^H(t)\| > d_{\text{cutoff}} \end{cases}$$

Thus, the number of contacts (*C*) for every residue (*i*) against each residue (*j*) in the remaining (*K*) helices are computed (Sica and Smulski 2021; Smulski et al., 2022).

$$C_{ij} = \sum_{t=t_0}^{T} \sum_{j}^{K \ne H} c_{ij}^{HK}(t)$$

*Note:* Normally cutoff distances between 0.8 and 1 nm provide a good representation of closely interacting residues. These values are comparable to the cut off distances used for computing van der Waals (0.9 nm) and Coulombic (1.2 nm) interactions (default values in the parametrization of the Martini force field - check these values in the file "production_310K.mdp"). With values closer to the bead diameter (0.47 nm for BB atoms) contact detection tends to zero. With values greater than 1.0 nm the algorithm increases the counts of contacts where BB atoms are not in direct interaction or have a third BB atom in the middle.

In this procedure, the contact detection along the simulation is performed by the GROMACS tool ``select''. You have to provide this tool with the number of the BB particle (*i*) and the number of all other BB particles of the rest of the peptides to check for possible contacts. This procedure is repeated for every BB particle. The number of particles (*i*) and particles (*j*) has to be listed in two groups in an index file (the syntax of this file is specific to GROMACS). In the procedure, you have to create one index file per particle (*i*). In our example, every peptide has 36 BB particles. Thus, for a system with 9 peptides you have to create 36 x (9 - 1) = 288 index files. GROMACS has a tool to create this type of files, but we use a Rscript to automate this step. Then we used the GROMACS tool ``select'' to compute the contacts. Finally, we process the output of the previous step with another Rscript to calculate the contact matrix. The matrix has dimension *R*x*R*, where (*R*) is the number of amino acids in the peptide (36 in this example).

18. Move to the first working directory (``PROT'', in this example) and copy the folder ``contact_map'' downloaded from the repository and move there. If you are in the folder "PROT'', use these commands:

```
> cp -r ../contact_maps/ ./
> cd ./contact_maps/
```

19. Create two new directories, one for the index files (``/indices'') and another for the output of the ``select'' command (``/output_contacts'').

```
> mkdir ./indices
> mkdir ./ouput_contacts
```

*Note:* The next steps generate a great number of temporary files.

20. Create the index files. First, copy the atom number of the BB particles from one (the first) peptide to a file named ``one_peptide_BB_atoms'' and atom number of the BB particles from all peptides to a file named ``all_peptides_BB_atoms''. We provide the next bash command to extract these numbers from two ".gro" files, which in our example are one step before in the directory tree.

```
# Assuming the correct path is ../. Change if necessary
> grep BB ../z_aligned_peptide.gro  | awk '{print $3}' >
./indices/one_peptide_BB_atoms
> grep BB ../peptide_array_mem.gro  | awk '{print $3}' >
./indices/all_peptides_BB_atoms
```

⚠ CRITICAL: Check the output to be sure the numbers are correct and no other line was copied.

21. Create a series of ``.ndx'' GROMACS index files. This step is performed with a Rscript named "indices.R" that you will find in the current directory.
    a. Open "indices.R" (you can use a text editor or RStudio) and edit the variables indicated in the Rscript as you need.
    b. Then, you can run the Rscript in RStudio or directly in the terminal with the next command.

```
> Rscript ./indices.R
```

22. Next, using the GROMACS command ``select'', compute the contacts between the BB particle in ``[ group1 ]'' and all other BB particles in ``[ group2 ]'', as defined in every ``.ndx'' file generated in the previous step (open any of the index files to verify the syntax). The output files of this step are saved in ``./ouput_contacts''. The GROMACS ``select'' command is executed with a bash script.
    a. Using a text editor, open the script ``contacts.sh'' located in this folder, edit the variables as you need: cutoff to filter interactions by distance, name of directories, etc.
    b. After editing your options, grant execution permission and run the script.

```
> chmod +x ./contacts.sh

> ./contacts.sh
```

*Note:* You can check ``gmx select -h'' for better understanding the options that we have selected and which other options you could use (i.e., initial and final time of analysis). For example, we used ``-pbc'' to detect contacts through periodic boundaries and ``-dt 100'' to control the time step of our analysis.

23. Use ``matrix.R'' to calculate how many times the particle "*BB(i)*" of any peptide contacted the particles "*BB(j)*" of the remaining peptides.
    a. Open this R script and edit your options as indicated (directories, number of atoms per peptide).
    b. Run the script in RStudio or in the terminal with the next command. The script will show you the number of BB particles per peptide and in the whole system, and will output the matrix in the shell. Troubleshooting 2.

```
> Rscript matrix.R
```

    c. If the script runs successfully two files are created: ``ContactMat.RDS'' and ``ContactMat.csv''. These files contain the same matrix of raw number of contacts in two different formats. Additionally, a figure ``Contact_map.png'' will be created with the normalized contact map in the format we used in our work (Figure 5).

*Note:* In some cases, it is useful to exclude the residues in the aqueous phase from the matrix graph. The script also produces the plot in "Contact_map_buried_residues.png" showing only the residues buried in the membrane.

## Radial and angular map analysis

Timing: 1 h

In this section we are going to analyze the structural organization of the contacts between peptides. We are going to build maps of radial distribution and angle orientation between peptides in the membrane *XY*-plane. In this analysis, we first calculate a centroid per peptide at the level of a selected residue. Then, every centroid in the array is considered a center of coordinates, and the position and orientation of the surrounding peptides along the simulation time is computed (Sica and Smulski 2021; Smulski et al., 2022).

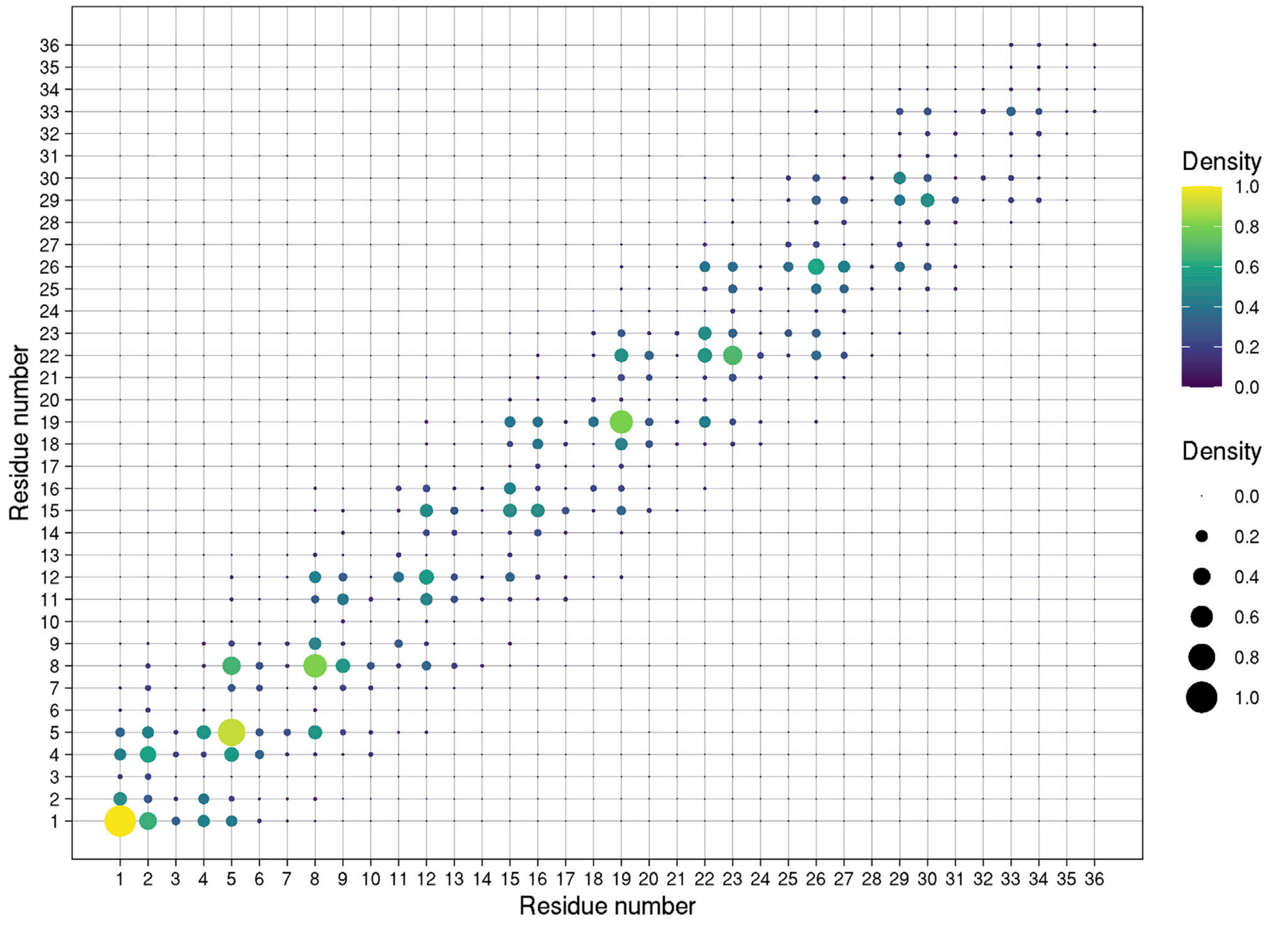


**Figure 5. Contact matrix**

"`Contact_map.png`" plot created with the normalized contact map for TACI. Each dot represents the normalized amount of contacts (Cij) between BB atoms of residues "i" (x axis) and "j" (y axis). To help visualization, the normalized value of Cij is represented by color (from yellow (high) to violet (low)) and size scales. Numbers correspond to each amino acid from the N-terminal region (1) to the c-terminal region (36).

To compute a centroid (*C*), we define a central residue and then take the coordinates of its BB particle (*i*) and of the BB particles of the previous and next residue in the sequence ($C(i) = [r(i-1)+r(i)+r(i+1)]/3$, where *r* is the *XYZ*-coordinate of the atom). Then, the unit bisection vector is computed between the central BB particle (*i*) and adjacent BB particles ($i \pm 1$). Finally, the centroid is considered the origin of the reference frame with its orientation vector aligned in the *X*-axis and the position and orientation of the centroids of the remaining peptides (8 in this example) are computed. This procedure is repeated for every peptide along a certain simulation time. The scatter plot of the accumulated *XY*-centroids positions and orientations are transformed to a density map with "`ggplot`" implemented in R.

In our analysis we are going to compute the next variables, which are illustrated in Figure 6:

***vdx*** and ***vdy***: Correspond to the *X*/*Y* location of the centroid of the peptide *hi* with respect to the centroid of the reference peptide (*href*).

***Alpha*** angle: Corresponds to the location of the centroid of the peptide *hi* with respect to the centroid of the reference peptide (*href*).

*Beta* angle: Corresponds to the orientation of the peptide *hi* with respect to the orientation of the reference peptide (*href*).

24. Return to the first directory ("PROT" in this example) and copy the folder ``radial_distribution'' downloaded from the repository and move there. If you are in the folder "PROT'', use these commands:

```
> cp -r ../radial_distribution/ ./

> cd ./radial_distribution/
```

25. Grant execution permission to the bash script ``extract_coordinates.sh''.

```
> chmod +x extract_coordinates.sh
```

The script performs the following three steps:

   a. First, the script creates an index file with the number of the BB particles necessary for computing the centroids of every peptide.
   b. Second, using this index and the GROMACS tool ``trjconv'', the script creates a trajectory in ``.gro'' format for these atoms.
   c. Finally, the script creates three different files with the values of the coordinates, box dimension and timestamp extracted from the trajectory file.

26. Edit the script ``extract_coordinates.sh''. In this example you only have to edit the paths to directories and files. If you are using a peptide different from the example, you will also need to modify the following variables:
   a. The variable ``FIRST_PEPTIDE_RESIDUE_UNIPROT'' corresponds to the Uniprot numeration of the first residue in the input PDB file. In our case this number is 156 (serine) in accordance with the Uniprot sequence O14836.
   b. The variable "CENTER_RESIDUE_UNIPROT" corresponds to the residue selected for the centroids calculation, according to Uniprot. In this example we used 174 (valine).
   c. Once edited, grant execution permission and run the script in the terminal:

```
> chmod +x extract_coordinates.sh

> ./extract_coordinates.sh
```

27. Use the Rscript ``radial_distribution.R'' present in this directory to perform the calculations of *vdx*, *vdy*, *alpha* and *beta* angles. Open this file and edit the variables. You can also follow the comments in the code to understand the calculations.

Run this Rscript in RStudio or in a terminal with the next command.

```
> Rscript ./radial_distribution.R
```

28. The script creates two files (``xyR_residueX.csv'' and ``xyR_residueX.RDS'') with identical data in different formats. The files contain the following data in columns:
   a. *Time*: time of the simulation.

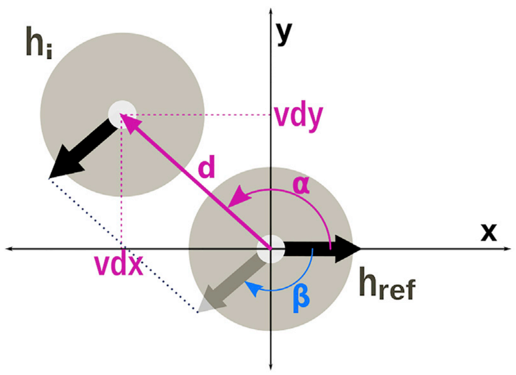


**Figure 6. System reference**
Scheme of the reference system used to calculate the relative positions and orientation of the helices when each helix is placed at the center (*X*=0, *Y*=0) of a virtual quadrant.

b. a,b: identification number of the peptides a and b interacting at distance less than cutoff.
c. *peptide_a*,*peptide_b*: peptide identification in letters code.
d. *distance*: geometric distance between centroids.
e. *vdx*,*vdy*: (*x*,*y*) location of peptide *b* with respect to peptide *a*.
f. *alpha*: *alpha* angle in degrees (-180 to 180) of the location of peptide *b* with respect to peptide *a*.
g. *beta*: angle between orientation vectors of peptide *b* with respect to peptide *a*.

29. The script also creates two graphs in ``.png'' format, as shown in Figure 7.
   a. ``Radial_map_residueXXX.png'': a density map of *vdx* vs. *vdy* for residue XXX accumulated along every interacting pair of peptides (*i*,*j*). The scatter plot is shown in the background.
   b. ``Angular_map_residueXXX.png'': a density map of alpha *angle* vs. *beta angle* for residue XXX accumulated along every interacting pair of peptides (*i*,*j*). The scatter plot is shown in the background.

## EXPECTED OUTCOMES

With this protocol you should be able to convert an all-atom structure of a transmembrane helix of TACI into a coarse-grained model (Figure 1), create an array of 9 transmembrane segments (Figure 2) and embed them in a lipid bilayer (Figure 3). Then, to perform a molecular dynamics simulation for a time-lapse of 3 microseconds (Figure 4) and finally to analyze the output of this simulation. The analysis was divided in three main axes:

First, contact matrix: this provides the main residues involved in helix-to-helix interaction by analyzing the whole system along the simulated period (Figure 5).

Second, radial map: this provides the main location of the interacting helices in the *X*/*Y* plane by defining a reference residue. As before, this analysis is performed on the whole system along the simulated period (Figure 7, left).

Third, angular map: using the same reference residue as the Radial map, this analysis provides the main location of the interacting helices in the *X*/*Y* plane but using the position in degrees around the central helix (*alpha*), combined with the relative orientation of the interacting pair (*beta*). As before, this analysis is performed on the whole system along the simulated period (Figure 7, right).

## LIMITATIONS

Molecular dynamic simulations usually need to be complemented with experimental evidence in order to generate meaningful results.

This kind of study requires powerful computational resources. This is particularly limiting when the simulation is performed with several transmembrane segments for several microseconds.

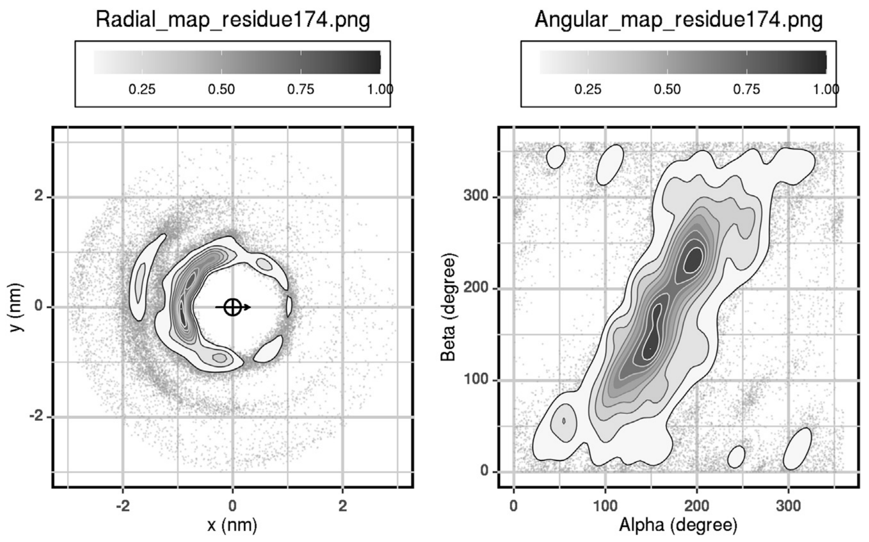


**Figure 7. Data analysis**
Graphs generated by ``radial_distribution.R'': ``Radial_map_residue174.png'' and ``Angular_map_residue174.png''. The center of the reference corresponds to V174.

The simulation time suggested in this protocol does not ensure the convergence of the system in a global minimum. Consider that the computation times necessary for convergence could be extremely long, depending on the characteristics of the system (protein and lipids).

Some peptides may interact very rarely, remaining mostly isolated along the simulation. In these cases, these analyses still produce an output and the figures of normalized values can render peaks corresponding to sporadic, unstable interactions that are not representative of the whole behavior, that is, that the peptides seldom interact. Therefore, it is likely that this approach is more reliable with stable interacting peptides than with unstable associations. Always, visually inspect the simulations with VMD or pymol.

The analysis of the data is in constant evolution in order to differentiate stable from unstable associations and to be able to produce a representative structure when possible.

## TROUBLESHOOTING

### Problem 1

Problem calling DSSP when using ``martinize.py'' (see "molecular model preparation" step 6).

### Potential solution

Try downloading the binary file that we provide in our repository (``./dssp/dssp-2.0.4-linux-i386''), and change the path after the ``-dssp'' flag in the previous command to that file. You can also try changing the file permissions using the following command:

```
> chmod +x dssp-2.0.4-linux-i386
```

If you still have problems, we provide a google colab's notebook that converts an all-atom model of the peptide into a coarse-grain MARTINI model. You can access the notebook with the following link: https://colab.research.google.com/drive/1Mewofw8z4Duai2MAJ_kc1LV_BsaGCbmZ#scrollTo=GAjOyyMSgO0k. This option also works for Windows users. To use the notebook all you have to do is upload the ``.pdb'' file you want to use (in our case ``peptide.pdb'') and ``martinize.py''. Then you have to run the notebook (you can use the command ``Crtl+F9'' to do it) and finally, download the generated files into the working directory (``PROT'' in this example).

### Problem 2

Error when running R scripts (see "contact map analysis" step 23 and "radial and angular map analysis" step 27). You may need to install specific packages.

### Potential solution

Open R in console.

```
> R
```

Then type.

```
> install.package(``ggplot2'')

> install.package(``reshape'')
```

Quit R.

```
> quit()
```

Alternatively:

You can install ``ggplot'' by running the following commands:

```
> sudo apt-get update -y

> sudo apt-get install -y r-cran-ggplot2
```

Also, you can install the ``reshape'' package using the following command:

```
> sudo apt-get install -y r-cran-reshape
```

## RESOURCE AVAILABILITY

### Lead contact

Further information and requests for resources and reagents should be directed to and will be fulfilled by the lead contact, Cristian R. Smulski (cristian.smulski@cab.cnea.gov.ar).

### Materials availability

This study does not use any materials.

### Data and code availability

The code used and generated during this study is available at https://gitlab.com/BioComp/taci_cg_simulations.

## ACKNOWLEDGMENTS

This work was supported by grants from Fondo para la investigación Científica y Tecnológica (FONCYT) PICT-2018-01107 / PICT-2019-00642 (to C.R.S.). M.V.K., M.P.S., and C.R.S. are supported by the national research council CONICET. We acknowledge Dr. P. Schneider and Dr. H.G. Asorey for their constant support. This work has been partially carried out on the ACME cluster, which is owned by CIEMAT and funded by the Spanish Ministry of Economy and Competitiveness project CODEC-OSE (RTI2018-096006-B-I00).

## AUTHOR CONTRIBUTIONS

Conceptualization, M.P.S. and C.R.S.; protocol elaboration, M.V.K., M.P.S., and C.R.S.; scripts development and repository, M.P.S.; scientific evaluation, C.R.S.; writing, review, and editing, M.V.K., A.A.M., M.P.S., and C.R.S.; funding acquisition, C.R.S.

## DECLARATION OF INTERESTS

The authors declare no competing interests.

## REFERENCES

Abraham, M.J., Murtola, T., Schulz, R., Páll, S., Smith, J.C., Hess, B., and Lindahl, E. (2015). GROMACS: high performance molecular simulations through multi-level parallelism from laptops to supercomputers. SoftwareX *1-2*, 19–25. https://doi.org/10.1016/j.softx.2015.06.001.

de Jong, D.H., Singh, G., Bennett, W.F.D., Arnarez, C., Wassenaar, T.A., Schäfer, L.V., Periole, X., Tieleman, D.P., and Marrink, S.J. (2013). Improved Parameters for the martini coarse-grained protein force field. J. Chem. Theory Comput. *9*, 687–697. https://doi.org/10.1021/ct300646g.

Humphrey, W., Dalke, A., and Schulten, K. (1996). VMD: visual molecular dynamics. J. Mol. Graph. *14*. 33–8– 27–8. https://doi.org/10.1016/0263-7855(96)00018-5.

Kabsch, W., and Sander, C. (1983). Dictionary of protein secondary structure: pattern recognition of hydrogen-bonded and geometrical features. Biopolymers *22*, 2577–2637. https://doi.org/10.1002/bip.360221211.

Marrink, S.J., Risselada, H.J., Yefimov, S., Tieleman, D.P., and de Vries, A.H. (2007). The MARTINI force field: coarse grained model for biomolecular simulations. J. Phys. Chem. B *111*, 7812–7824. https://doi.org/10.1021/jp071097f.

Monticelli, L., Kandasamy, S.K., Periole, X., Larson, R.G., Tieleman, D.P., and Marrink, S.J. (2008). The MARTINI coarse-grained force field: Extension to proteins. J. Chem. Theory Comput. *4*, 819–834. https://doi.org/10.1021/ct700324x.

Sica, M.P., and Smulski, C.R. (2021). Coarse grained molecular dynamic simulations for the study of TNF receptor Family Members' transmembrane organization. Front. Cell Dev. Biol. *8*, 577278–577311. https://doi.org/10.3389/fcell.2020.577278.

Siewert, J.M., Corradi, V., Souza, P.C.T., Ingólfsson, H.I., Tieleman, D.P., and Sansom, M.S.P. (2019). Computational modeling of realistic cell membranes. Chem. Rev. *119*, 1–43. https://doi.org/10.1021/acs.chemrev.8b00460.

Smulski, C.R., Zhang, L., Burek, M., Teixidó Rubio, A., Briem, J.S., Sica, M.P., Sevdali, E., Vigolo, M., Willen, L., Odermatt, P., et al. (2022). Ligand-independent oligomerization of TACI is controlled by the transmembrane domain and regulates proliferation of activated B cells. Cell Rep. *38*, 110583. https://doi.org/10.1016/j.celrep.2022.110583.

Touw, W.G., Baakman, C., Black, J., te Beek, T.A.H., Krieger, E., Joosten, R.P., and Vriend, G. (2015). A series of PDB-related databanks for everyday needs. Nucleic Acids Res. *43*, D364–D368. https://doi.org/10.1093/nar/gku1028.

Wassenaar, T.A., Ingólfsson, H.I., Böckmann, R.A., Tieleman, D.P., and Marrink, S.J. (2015). Computational Lipidomics with insane: a versatile tool for generating custom membranes for molecular simulations. J. Chem. Theory Comput. *11*, 2144–2155. https://doi.org/10.1021/acs.jctc.5b00209.