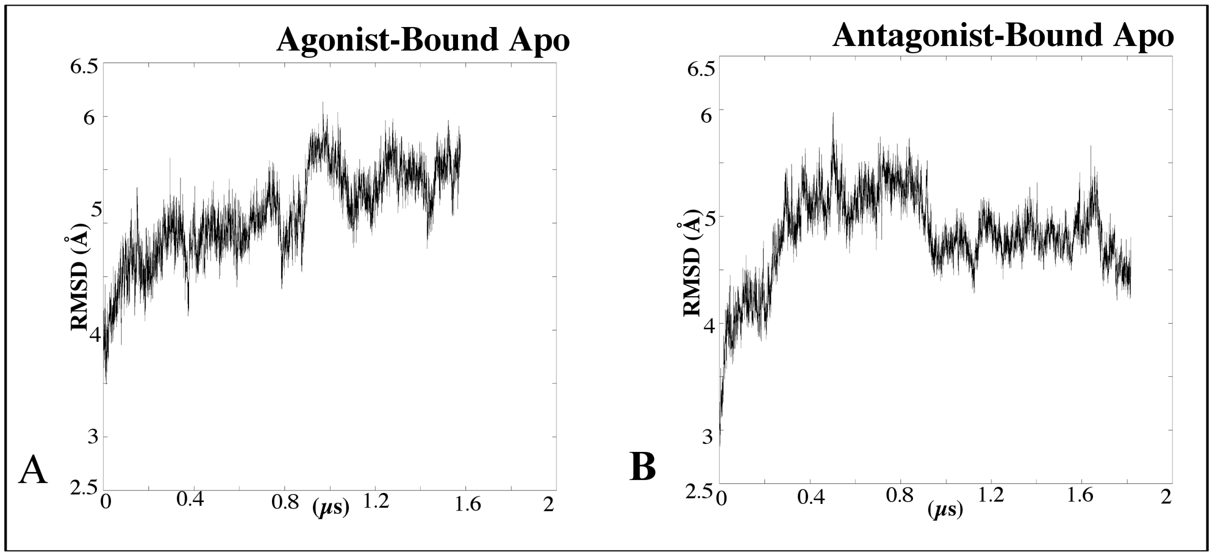
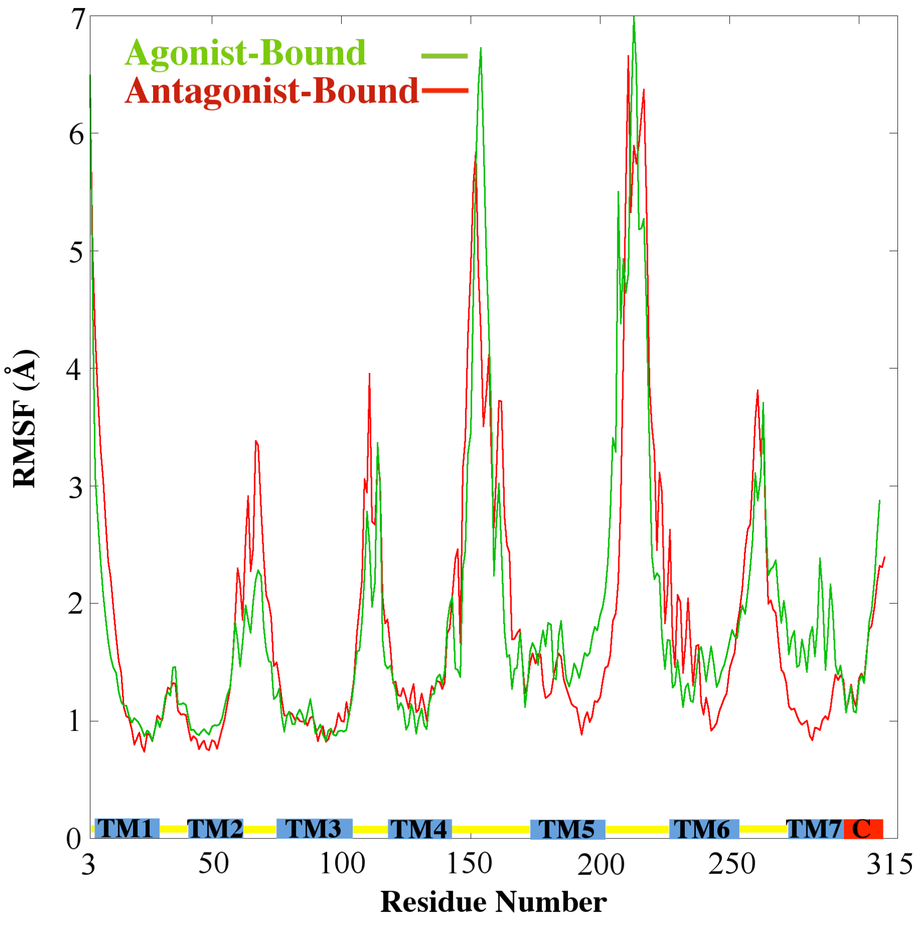
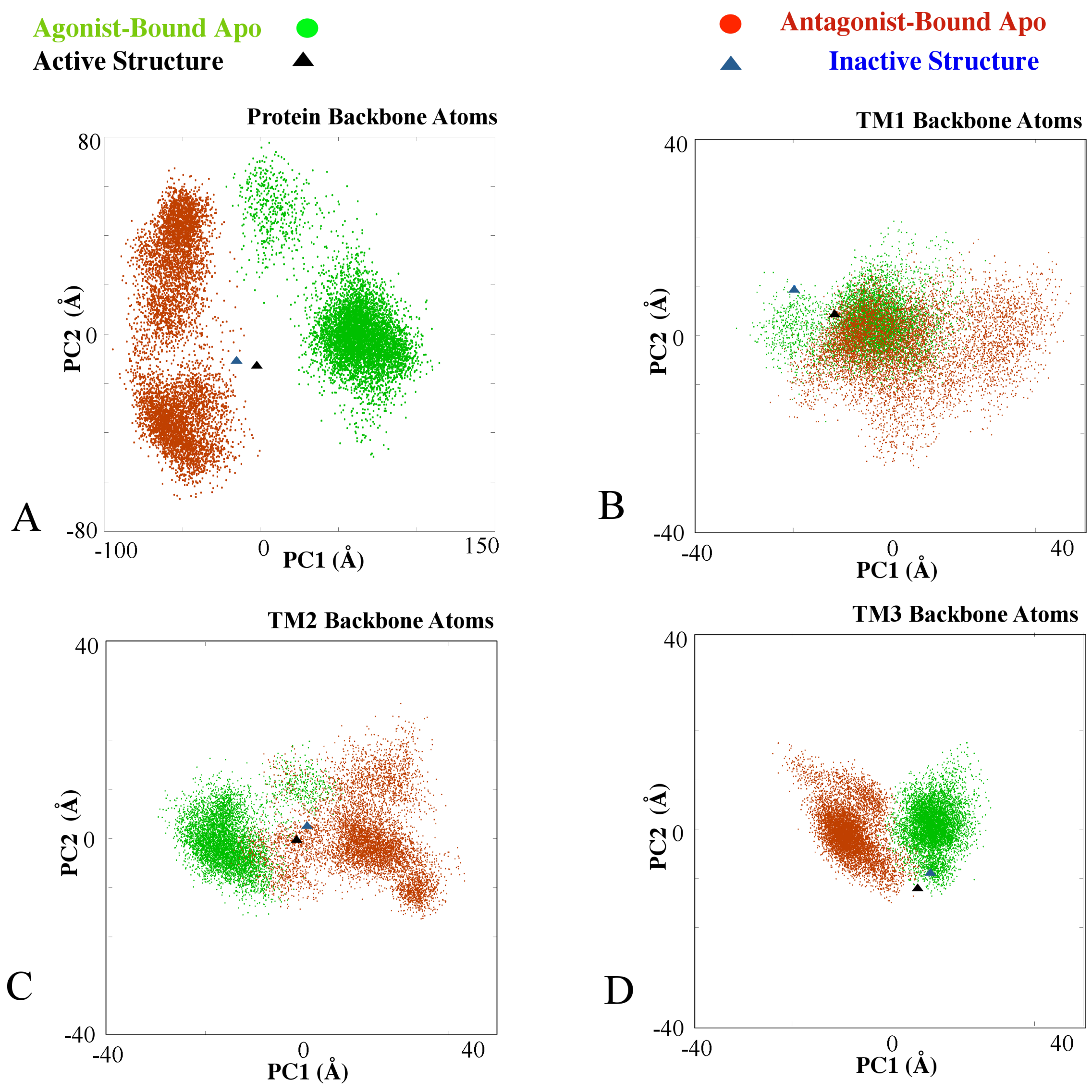
Supp. Fig. 1 The RMSD of Cα atoms to the respective starting structures. (A) Shows the antagonist-bound apo simulation, while (B) shows the initial agonist-bound apo simulation in green.



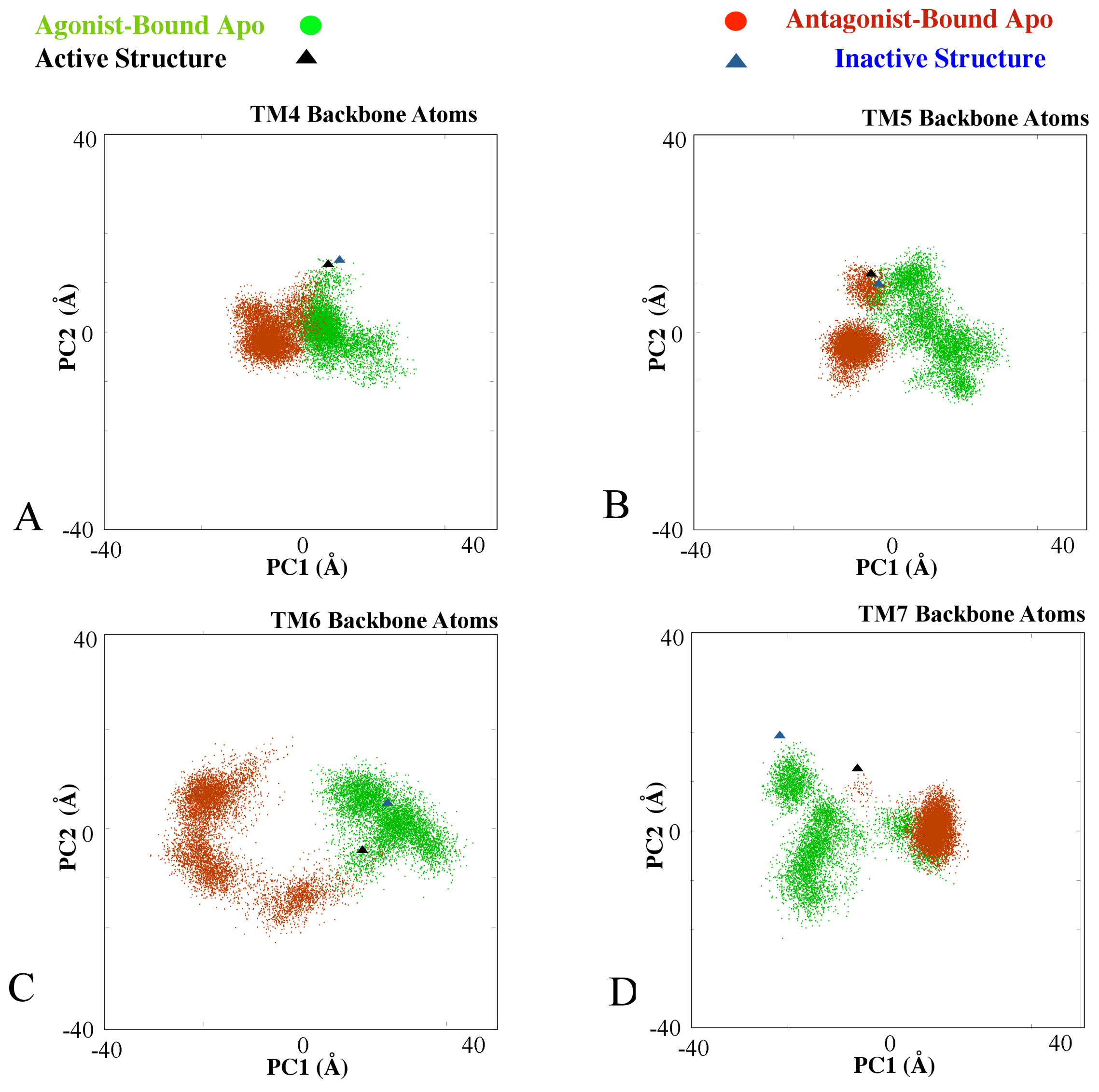
Supp. Fig 2 RMSF of Cα : the initial antagonist-bound apo simulation in red and the initial agonist-bound apo simulation in green. The blue bars represent the location of TM, while the yellow bars represent the location of the loops. The C-Term is shown as a red bar.



Supp. Fig 3 Principal component analysis of all the backbone atoms (A), the backbone atoms of TM1 (B), the backbone atoms of TM2 (C), and the backbone atoms of TM3 (D), for both simulations with the starting active structure shown as a black triangle, the starting inactive structure as a blue triangle, the initial antagonist-bound apo simulation shown as red dots, and the initial agonist-bound apo simulation shown as green dots.

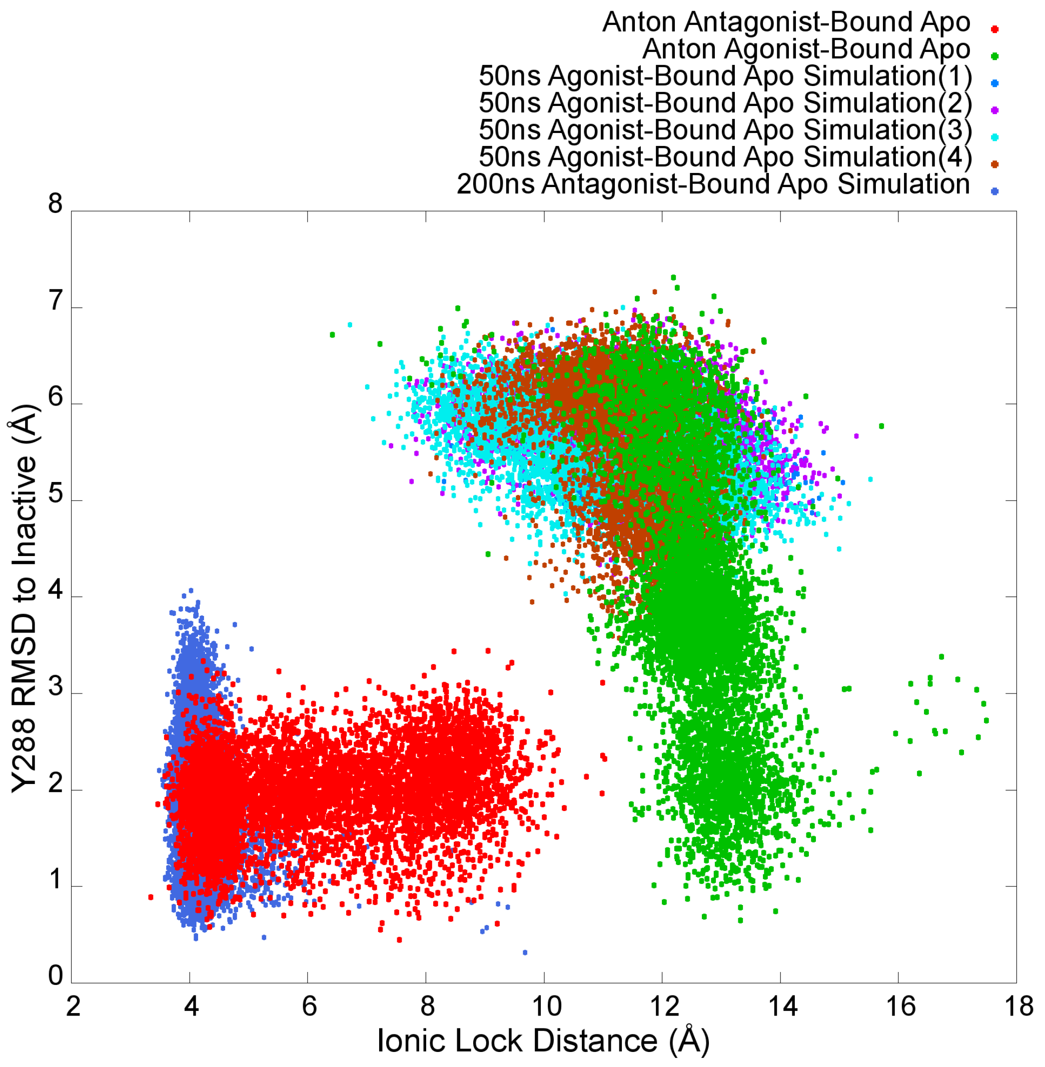


Supp. Fig 4 Principal component analysis of the backbone atoms of TM4 (A), the backbone atoms of TM5 (B), the backbone atoms of TM6 (C) and the backbone atoms of TM6 (D) for both simulations with the starting active structure shown as a black triangle, the starting inactive structure as a blue triangle, the initial antagonist-bound apo simulation shown as red dots, and the initial agonist-bound apo simulation shown as green dots.



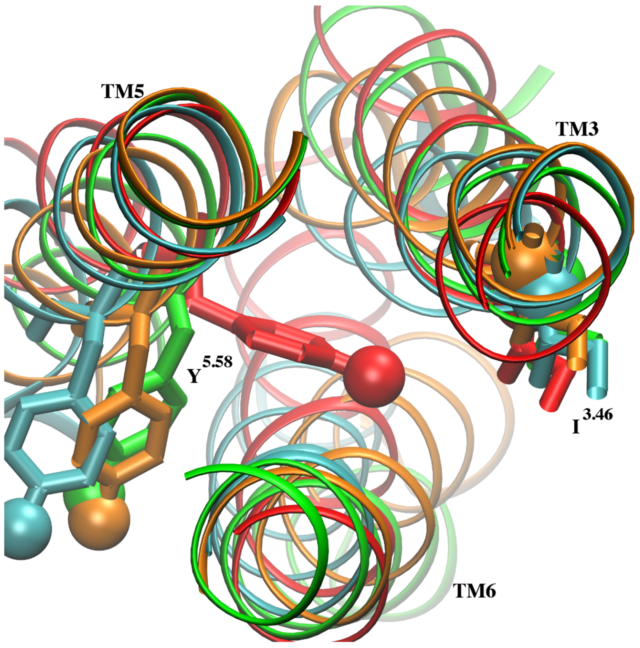
Supp. Fig 5

The distance between the ionic lock (R102$^{3.50}$ - E228$^{6.30}$) and the RMSD of Y288$^{7.53}$ in the NPxxY motif relative to the inactive structure is plotted for the five control simulations, one 200ns antagonist-bound apo simulation (dark blue) and four 50ns agonist-bound apo simulations (light blue, purple, turquoise, brown), and the Anton antagonist-bound apo simulation (red) and agonist-bound apo simulation (green).

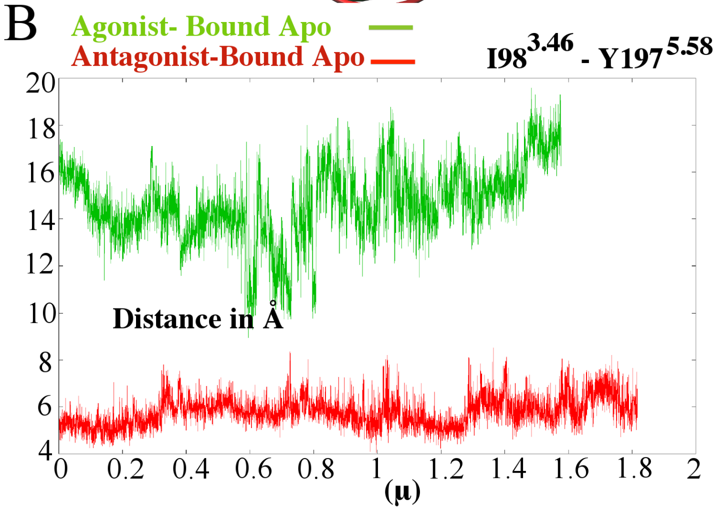


Supp Fig. 6 Conformation of Y197$^{5.58}$ during initial agonist-bound apo and initial antagonist-bound apo simulations: (A) Cartoon representation of the conformation of Y197$^{5.58}$. Y197$^{5.58}$ and I98$^{3.46}$ are shown as sticks, and the atoms of Y197$^{5.58}$ (hydroxyl oxygen) and I98$^{3.46}$ (Cα atom) used to calculate distance are as balls. The aligned active (green), inactive (red), intermediate one (blue) and intermediate two (orange) structures are shown in cartoon. (B) The calculated distance between I98$^{3.46}$ and Y179$^{5.58}$ in the initial antagonist-bound Apo simulation (red), and the initial agonist-bound apo simulation (green.)

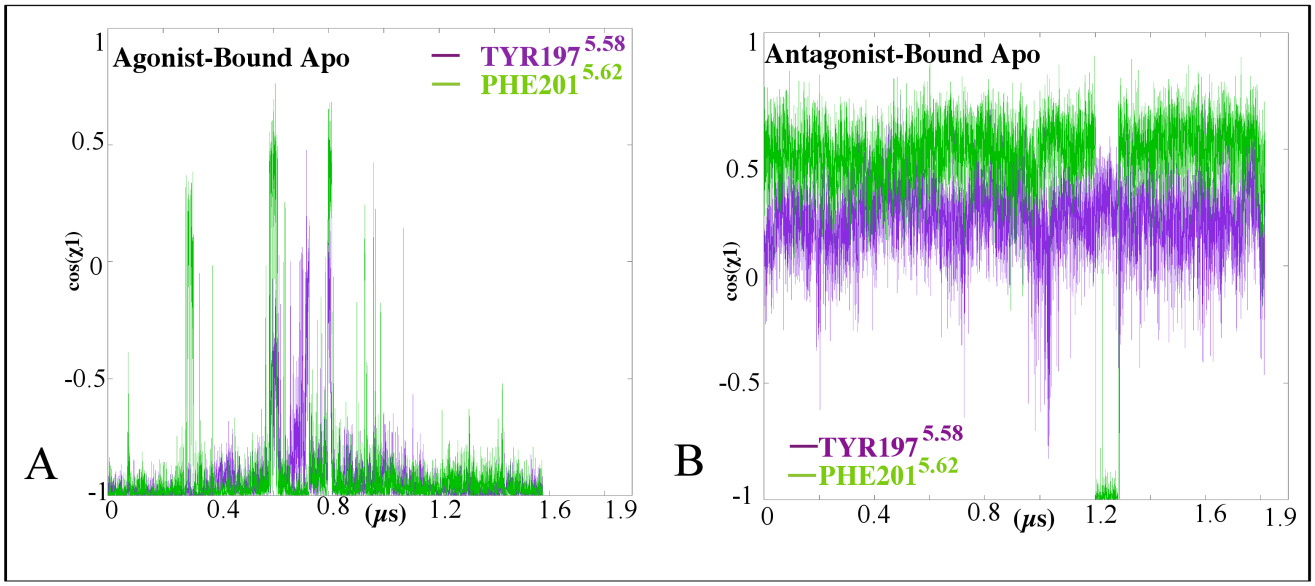
A



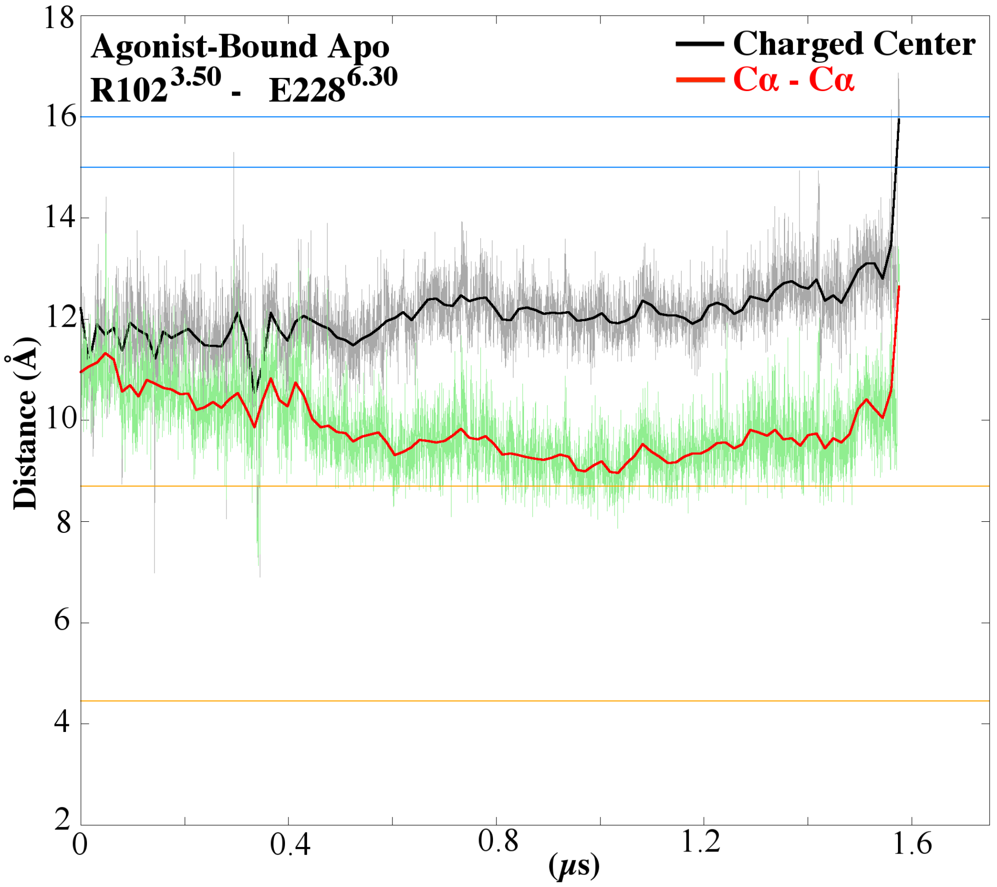
B



Supp. Fig 7 The side-chain dihedral $\chi_1$of Y197$^{5.58}$ and F201$^{5.62}$ (A) in the initial antagonist bound apo simulation and (B) the initial agonist-bound apo simulation.



Supp. Fig. 8 Distance between the R102$^{3.50}$ – E228$^{6.30}$ ionic lock in the initial agonist-bound apo simulation: The smoothed distance between charged centers of R102$^{3.50}$ and E228$^{6.30}$ is plotted in black and the distance between the Cα atoms in red, while the raw data in gray and pink, respectively. The distance between the charged center and Cα for inactive rhodopsin (4.5Å and 8.6Å, respectively), shown as orange lines (PDB ID: 1F88) [1] . The distance between the charged center and Cα for active rhodopsin (16Å and 14.6Å, respectively), shown as blue lines (PDB ID: 3CAP) [2].



1. Palczweski, K., et al., *Crystal structure of rhodopsin: a G protein-coupled receptor.* Science, 2000. **289**(5480): p. 739-745.
2. Park, J.H., et al., *Crystal structure of the ligand-free G-protein-coupled receptor opsin.* Nature, 2008. **454**(7201): p. 183-187.

Convert Namd2Maestro

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
############################################################
#
# this script converts a snapshot from a namd trajectory
# to a Maestro input file.
#
# (C) 2010 Markus Dittrich, NRBSC, PSC, CMU
#
# call with:
#
# vmd -dispdev text -python -e convertNAMDtoMaestro.py \
#     -args -p ubq_wb.psf \
#     -c ubq_wb_eq.restart.coor -v ubq_wb_eq.restart.vel \
#     -x ubq_wb_eq.xsc -o outfile -s -S "protein"
#
# This program is distributed in the hope that it will be useful,
# but WITHOUT ANY WARRANTY; without even the implied warranty of
# MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
#
############################################################

import sys
import optparse
from atomsel import *
from AtomSel import AtomSel
from Molecule import Molecule
from VMD import evaltcl


def parse_cmdline(cmdlineArgs):
  """
  This function initializes the command line parser.
  """

  parser = optparse.OptionParser("Usage: vmdt -python -e "
        "readVelocities.py -args [options]")

  parser.add_option("-p", "--psffile", action="store", dest="psfFile")
  parser.add_option("-c", "--coorfile", action="store", dest="coorFile")
  parser.add_option("-v", "--velfile", action="store", dest="velFile")
  parser.add_option("-x", "--xscfile", action="store", dest="xscFile")
  parser.add_option("-o", "--outputfile", action="store", dest="outFile")
  parser.add_option("-s", "--centerSystem", action="store_true", dest="doCenter")
  parser.add_option("-S", "--centerSelection", action="store", dest="centerSel")
```

```
    parser.set_defaults(doCenter = False, centerSel = "all")

    opts, args = parser.parse_args(cmdlineArgs)
    psfFile   = opts.psfFile
    coorFile  = opts.coorFile
    velFile   = opts.velFile
    xscFile   = opts.xscFile
    outFile   = opts.outFile
    doCenter  = opts.doCenter
    centerSel = opts.centerSel

    # all filenames are required
    if (psfFile == None) or (coorFile == None) or (velFile == None) \
      or (xscFile == None) or (outFile == None):

            parser.print_help()
            exit()


    return psfFile, coorFile, velFile, xscFile, outFile, doCenter, \
        centerSel



def load_velocities(psfFile, velFile):
    """
    Load the binary velocity file and extract velocities.
    """

    mol = Molecule()
    mol.load(psfFile)
    mol.load(velFile, "namdbin")

    allVelocities = atomsel('all')
    xVel = allVelocities.get('x')
    yVel = allVelocities.get('y')
    zVel = allVelocities.get('z')

    # conversion from binvel units to A/ps
    convFactor = 20.4582651391
    xVel = [v * convFactor for v in xVel]
    yVel = [v * convFactor for v in yVel]
    zVel = [v * convFactor for v in zVel]

    mol.delete()
    return xVel, yVel, zVel



def load_system(psfFile, coorFile):
    """
    Load the main system.
    """

    mol = Molecule()
    mol.load(psfFile)
    mol.load(coorFile, "namdbin")
    return mol



def set_velocities(mol, xVel, yVel, zVel):
    """
    Add the molecule velocities to the system.
    """

    allAtoms = atomsel("all")
    allAtoms.set("vx", xVel)
    allAtoms.set("vy", yVel)
    allAtoms.set("vz", zVel)



def save_mol_as_maestro(mol, fileName):
    """
    Save the current molecule as maestro file.
    """

    mol.save(fileName + ".mae")



def set_pbc(xscFile):
    """
    Sets the systems periodic boundaries."
    """
    xscFile = open(xscFile,"r")

    for line in xscFile:
        continue
    items = line.split()
    xDim  = items[1]
    yDim  = items[5]
    zDim  = items[9]

    #set pbd
    pbcCommand = ("package require pbctools; pbc set { %s %s %s }"
            % (xDim, yDim, zDim))
    evaltcl(pbcCommand)

    xscFile.close()



def center_system(selection):
    """
    Center the system around the selection.
    """

    centerSel = atomsel(selection)
    center = centerSel.center()
    negCenter = [-1.0 * item for item in center]

    moveSel = atomsel("all")
    moveSel.moveby(negCenter)



def remove_tip3p_hh_bond():
    """
    This removes the bond between hydrogen atoms in
    TIP3P water if present since viparr will introduce
    the proper constraint.
    """

    # it looks like atomsel doesn't support set/getbonds
    # so we have to use the deprecated AtomSel for now
    oh2Sel = AtomSel("resname TIP3 and name OH2", 1)
    h1Sel  = AtomSel("resname TIP3 and name H1", 1)

    oh2Indices = oh2Sel.get("index")
    bondlist = []
    for i in oh2Indices:
        bondlist.append([i])
    h1Sel.setbonds(bondlist)
```

```
#######################################################
# main routine
#######################################################
if __name__ == "__main__":

   # parse the command line
   psfFile, coorFile, velFile, xscFile, outfile, doCenter, \
      centerSel = parse_cmdline(sys.argv[1:])

   # transform NAMD to Maestro
   vx, vy, vz = load_velocities(psfFile, velFile)
   mol = load_system(psfFile, coorFile)

   if doCenter:
      center_system(centerSel)

   set_velocities(mol, vx, vy, vz)
   set_pbc(xscFile)
   remove_tip3p_hh_bond()
   save_mol_as_maestro(mol, outfile)

   exit()
```