INFORMAÇÕES CLIMÁTICAS CUSTOMIZADAS PARA OS EIXOS DE AÇÃO DO PROADAPTA

Produto 1 — Repositório de Dados Climáticos

Felipe Ferreira Alexandre

Por ordem do Ministério Federal do Meio Ambiente, Proteção da Natureza e Segurança Nuclear da República Federal da Alemanha

Por meio da giz Deutsche Gesellschaft für Internationale Zusammenarbeit (GIZ) GmbH

PROADAPTA Adaptação à Mudança do Clima

INPE

MINISTÉRIO DO MEIO AMBIENTE

PÁTRIA AMADA BRASIL GOVERNO FEDERAL

Elaborado por:
Felipe Ferreira Alexandre

Este documento foi produzido por consultores independentes no âmbito da implementação do Projeto Apoio ao Brasil na Implementação da sua Agenda Nacional de Adaptação à Mudança do Clima (ProAdapta).

O ProAdapta é fruto da parceria entre o Ministério do Meio Ambiente do Brasil (MMA) e o Ministério Federal do Meio Ambiente, Proteção da Natureza e Segurança Nuclear (BMU, sigla em alemão), no contexto da Iniciativa Internacional para o Clima (IKI, sigla em alemão) e implementado pela Deutsche Gesellschaft für Internationale Zusammenarbeit (GIZ) GmbH.

Contribui para o alcance dos objetivos deste projeto e para a coordenação técnica, em parceria com a GIZ, do processo de origem deste documento, o Instituto Nacional de Pesquisas Espaciais (INPE), por meio do Centro de Ciência do Sistema Terrestre (CCST)

Todas as opiniões aqui expressas são de inteira responsabilidade dos autores, não refletindo necessariamente a posição da GIZ, do INPE e do MMA. Este documento não foi submetido à revisão editorial.

MMA
Secretaria de Clima e Relações Internacionais
Departamento de Clima

GIZ
Ana Carolina Câmara (Coordenação)
Eduarda Freitas
Pablo Borges

INPE
Gean Pierre Henry Balbaud Ometo
Lincoln Muniz Alves

Ministério do Meio Ambiente
Esplanada dos Ministérios, Bloco B, Brasília/DF, CEP 70068-901
Telefone: + 55 61 2028-1206

Instituto Nacional de Pesquisas Espaciais
Av. dos Astronautas, 1758 Jardim da Granja São José dos Campos-SP, CEP 12227-010
Telefone: +55 (12) 3208-7776

Deutsche Gesellschaft für Internationale Zusammenarbeit (GIZ) GmbH
Sede da GIZ: Bonn e Eschborn
GIZ Agência Brasília
SCN Quadra 01 Bloco C Sala 1501
Ed. Brasília Trade Center 70.711-902 Brasília/DF
T + 55-61-2101-2170
E giz-brasilien@giz.de
www.giz.de/brasil

A encargo de:
Ministério Federal do Ambiente, Proteção da Natureza e Segurança Nuclear (BMU) da Alemanha
BMU Bonn:
Robert-Schuman-Platz 3 53175 Bonn, Alemanha
T +49 (0) 228 99 305-0

Diretora de Projeto:
Ana Carolina Câmara
T:+55 61 9 99 89 71 71
T +55 61 2101 2098
E ana-carolina.camara@giz.de

Brasília, janeiro de 2021

Projeto: Apoio ao Brasil na Implantação da Agenda Nacional de Adaptação à Mudança do Clima - PROADAPTA
PN: 15.9060.3-001.00

Termo de Referência
Informações Climáticas Customizadas para os eixos de ação do ProAdapta

Produto 1: Repositório de dados climáticos

Cooperação Alemã para o Desenvolvimento Sustentável – GIZ no Brasil
Janeiro – 2021
Versão 01.0

Consultor: Felipe Ferreira Alexandre

RESUMO

A mudança do clima tem e terá impactos diretos e indiretos sobre sistemas naturais, grupos e sistemas humanos, assim como sobre a atividade econômica. Focando nessa premissa o presente documento refere-se a criação de um repositório de dados climáticos com o propósito de subsidiar a análise do risco climático no âmbito das linhas de ação do ProAdapta. Para isto, foram utilizados dados climáticos de precipitação e temperatura do estado-da-arte de modelos climáticos globais do Painel Intergovernamental sobre Mudanças Climáticas (IPCC) - Projeto de Intercomparação de Modelos Acoplados Fase 6 (CMIP6), modelos climáticos regionais participantes do Coordinated Regional Climate Downscaling Experiment (CORDEX), e dados observacionais sobre a América do Sul e Central O repositório inlcui dados observacionais do CHIRPS e CPC e dados de modelos de clima das bases do CMIP6 e do CORDEX. As variáveis são temperatura máxima, temperatura mínima e precipitação com resolução temporal diário e o recorte da área de domínio é a America do Sul. A base conta com mais de 30 modelos que, combinados com os diferentes cenários de emissões de GEE (baixas e altas emissões), totalizam um conjunto de 90 projeções de clima na escala global e 66 na escala regional, ambos, para um horizonte temporal até 2100. O repositório tem 2,8 Terabytes de tamanho e está disponível para uso. O produto apresentado tem como objetivo permitir aos projetos, acesso as diferentes fontes de dados atualizadas, organizadas e de fácil acesso para extrair as informações para tratamento e desenvolvimento na pesquisa.

Sumário

1. Introdução

Em reação aos efeitos adversos da mudança do clima que impactam os sistemas naturais, humanos, produtivos e de infraestrutura, o governo brasileiro desenvolve uma agenda de adaptação voltada à gestão e à diminuição do risco climático do país, tendo o Plano Nacional de Adaptação (PNA) como o principal instrumento político.

Nesse contexto, o projeto “Apoio ao Brasil na Implantação da Agenda Nacional de Adaptação à Mudança do Clima - PROADAPTA” visa favorecer o aumento da resiliência climática do Brasil, por meio da implementação efetiva da Agenda Nacional de Adaptação, mediante apoio a processos de coordenação e cooperação entre as três esferas de governo, setores econômicos e sociedade civil, uma vez que os impactos da mudança do clima ocorrem em escala local, mas as medidas de enfrentamento dependem de ações coordenadas e implementadas em diferentes estratégias setoriais ou temáticas.

Avaliações de risco dos impactos da mudança do clima são cruciais para auxiliar em medidas de adaptação efetivas e garantir uma sociedade mais resiliente. Para tais análises, diversos conjuntos de dados e produtos de clima estão disponíveis. No entanto, o uso destas informações ainda é uma tarefa desafiadora. O emergente campo de Serviços Climáticos visa preencher essa lacuna através da customização de informações climáticas para os usuários finais tornando-se passo fundamental para a consolidação de análises de risco climático efetivas. Por esta razão, o projeto ProAdapta fomenta o desenvolvimento de serviços climáticos para suas diversas linhas de ação, tais como iniciativas do Ministério de Infraestrutura, Movimento Viva Água (MVA) conduzida pela Fundação Grupo Boticário, Defesa Civil de Santa Cataria, e Agência Nacional de Transportes Aquaviários.

2. Objetivo

O objetivo dessa etapa do trabalho foi a criação de um repositório de dados climáticos que contém o estado-da-arte de modelos climáticos globais e regionais, e dados observados possibilitando um ambiente que seja adequado e eficiente para acesso, manipulação e compartilhamento de dados entre vários usuários e aplicações no âmbito do projeto ProAdapta e também para plataforma “Projeções Climáticas no Brasil” do INPE.

3. Materiais e Métodos

A definição de um repositório de dados implica inicialmente especificar os tipos e as estruturas (formatos e frequência) dos dados a serem armazenados. Nesse sentido, a primeira etapa do trabalho foi a caracterização dos dados, que significa conhecer as características dos diferentes conjuntos de dados disponíveis. As demais etapas são a coleta dos dados seguido pela padronização e organização dos dados que alimentarão o repositório.

Dados climáticos correspondem a uma massa de informações sobre o clima oriundos de diversas variáveis e disponibilizadas ao longo do tempo por diversas instituições nacionais e internacionais. A dificuldade em acessar, gerenciar e manipular essas informações estão no fato de que apesar de estarem num portal único, a exemplo do portal do *Earth System Grid Federation* (ESGF), este constitui-se em um ferramenta complexa para usuários não técnicos e assim uma das grandes barreiras para acesso as informações relatadas pelos usuários.

3.1. Seleção de dados

Visando a melhor representação possível do clima atual e futuro, nesse projeto foram selecionados dados climáticos oriundos do *Climate Hazards group Infrared Precipitation with Stations* (CHIRPS) e *Climate Prediction Center* (CPC) obtidos a partir de observações locais e remotas, que representam os dados observados. As projeções foram selecionadas do Projeto de Intercomparação de Modelos Acoplados Fase 6 (CMIP6) e do *Coordinated Regional Climate Downscaling Experiment* (CORDEX).

A base de dados extraída das diferentes fontes limitou-se as as variáveis de interesse neste trabalho, são elas: precipitação, temperatura máxima e mínima, com frequência temporal diária e formato NetCDF (Network Common Data Form).

3.1.1 Dados observacionais

No que concerne a base de dados observacional os dados usados como referência tiveram como base os dados de precipitação diária do NOAA Climate Prediction Center (CPC), com resolução espacial 0.5° x 0.5° latitude por longitude, a partir da interpolação e controle de qualidade dos dados de precipitação reportados por aproximadamente 30.000 estações através do sistema Global Telecommunication System (GTS), além de outras fontes de dados de instituições nacionais tais como, Agência Nacional de Energia (ANEEL), Agência Nacional de Águas (ANA) e Centros Regionais de Meteorologia. Os dados podem ser obtidos no site

https://psl.noaa.gov/data/gridded/data.cpc.globaltemp.html

Já o CHIRPS é um conjunto de dados de precipitação quase global com mais de 35 anos. Abrange a faixa de latitude 50°S-50°N (e todas as longitudes) e variando de 1981 até quase o presente. O CHIRPS incorpora, imagens de satélite com resolução de 0,05 ° e dados de estação in-situ para criar séries temporais de chuva em grade para análise da variabilidade climática. Os dados podem ser obtidos no site https://data.chc.ucsb.edu/products/CHIRPS-2.0/

Estes dados do CPC e CHIRPS têm sido usados em vários estudos de mudanças climáticas sobre a América do Sul. Os dados foram detalhados sobre a América do Sul (Figura 1), e foi selecionada por constituir foco de importantes pesquisas, o que reflete a relevância destas áreas para os estudos do bioma floresta tropical, dos sistemas climáticos, hidrológicos e sociais associados a elas.

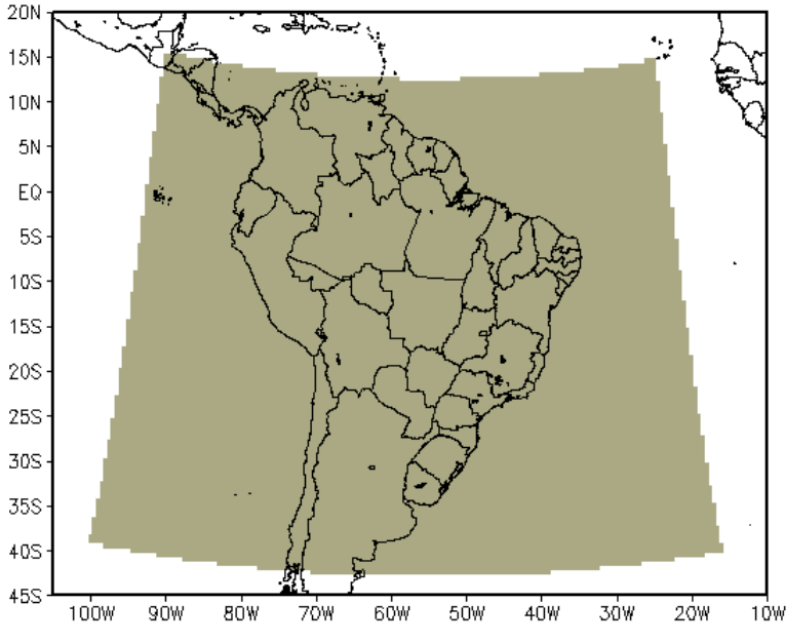

Figura 1 – Destaque da área selecionada

3.1.2 Projeções de modelos de clima

Diversos centros de pesquisa conduzem experimentos de modelagem do clima. A iniciativa *Coupled Model Intercomparison Project* (CMIP) tem por objetivo coordenar os esforços globais em modelagem de clima entorno de um protocolo de experimentos que permite a comparação entre os resultados dos modelos e seu uso em conjunto (*multi-model ensemble*). Maiores informações sobre o CMIP podem ser obtidas através do portal https://www.wcrp-climate.org/wgcm-cmip/wgcm-cmip6. Esse contrato considerou as projeções dos modelos de clima globais mais atuais, os quais compõem a fase 6 do

CMIP (CMIP6) e que farão parte do próximo relatório do IPCC, o *Sixth Assessment Report* (AR6). No que tange as projeções climáticas do CMIP6 os cenários selecionados foram o *Shared Socioeconomic Pathways* (SSPs, sigla em inglês) – SSP1.26 E SSP5.85 desenvolvidos em preparação ao próximo relatório do IPCC e representam possíveis mundos futuros que diferem em sua população, crescimento econômico, demanda de energia, igualdade e outros fatores. O download da base de dados encontra-se disponível no portal http://esgf-node.llnl.gov/

Em muitos casos, projeções em escala regional são preferíveis, o que implica numa melhor discretização dos eventos climáticos (espacial e temporalmente). Similar ao CMIP, o CORDEX cordena experimentos com modelos regionais de clima permitindo assim a comparação e uso em conjunto. O CORDEX é dividido em regiões e, nesse caso, foi selecionado a área denominada como SAM (South AMerica) apresentado na figura 2. Maiores informações sobre o CORDEX podem ser acessadas no portal do projeto https://cordex.org/ e o download de sua base de dados através do portal ESGF (https://esg-dn1.nsc.liu.se/projects/cordex/).

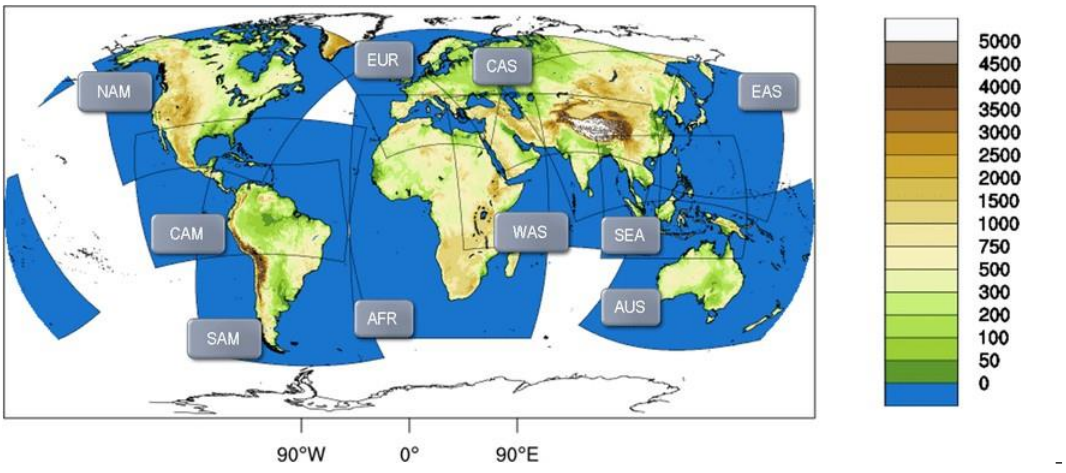

Figura 2 – Domínios do projeto *CORDEX*. **Fonte:** (REMEDIO et al., 2019)

Para as projeções climáticas do CORDEX os cenários selecionados foram o Representative Concentration Pathways (RCPs, sigla em inglês) – RCP4.5 e RCP8.5 desenvolvidos pelo IPCC trazendo uma tragetória na concentração de gases do efeito estufa. O RCP 4.5 é um cenário intermediário em que o forçamento radiativo está estabilizado a aproximadamente 4,5Wm^2 e o RCP 8.5 é um patamar elevado para forçamento radiativo superior a 8,5 Wm^2.

3.2. Pré-processamento de dados

Cada modelo de clima utiliza um sistema próprio de coordenadas. O primeiro passo do processo foi a padronização desses sistemas de coordenadas. Para tanto, realizou-se um pré-processamento para que toda a base estivesse na mesma resolução horizontal, isto é, uma interpolação bilinear para resolução de 1º x 1º latitude/longitude, para essa etapa foi utilizada a função *remapbil* do Climate Data Operators (CDO, https://code.mpimet.mpg.de/projects/cdo/), Assim o repositório foi padronizado em sua resolução espacial. Importante destacar que a resolução horizontal original dos dados dos modelos globais do CMIP6 e regionais foram mantida, caso exista alguma requisição futura que dependa da resolução original de algum modelo.

A obtenção de todos os dados foram feitos em ambiente Linux, utilizando scripts em linguagem shell um exemplo pode ser visto no (Anexo 1) que contem o script utilizado na obtenção dos dados do modelo BCC-CSM2-MR para o experimento histórico. Além da padronização da resolução horizontal realizou-se também o recorte do domínio definido para os mapas no repositório, que pode ser visto na figura 2. Esse processo pode ser visto no (Anexo 2). Outro passo desenvolvido na manipulação dos dados foi a conversão das unidades das variáveis obtidas. A precipitação foi convertida de $Kg/m^2/s$ para milímetro/dia e as temperaturas máxima e mínima de Kelvin para graus Celsius.

4. Repositório

Na visão do gerenciador de arquivos do sistema operacional windows, a estrutura hierárquica ficou definida pelo diretório raiz nominado de Repositório, seguido do nome das bases de dados, CHIRPS, CMIP6, CORDEX e CPC, como exibido na figura 3. No caso do CHIRPS e CPC, como são dados observados, não possuem mais diretórios dentro deles, apenas os dados padronizados com o domínio definido que pode ser visto na figura 2.

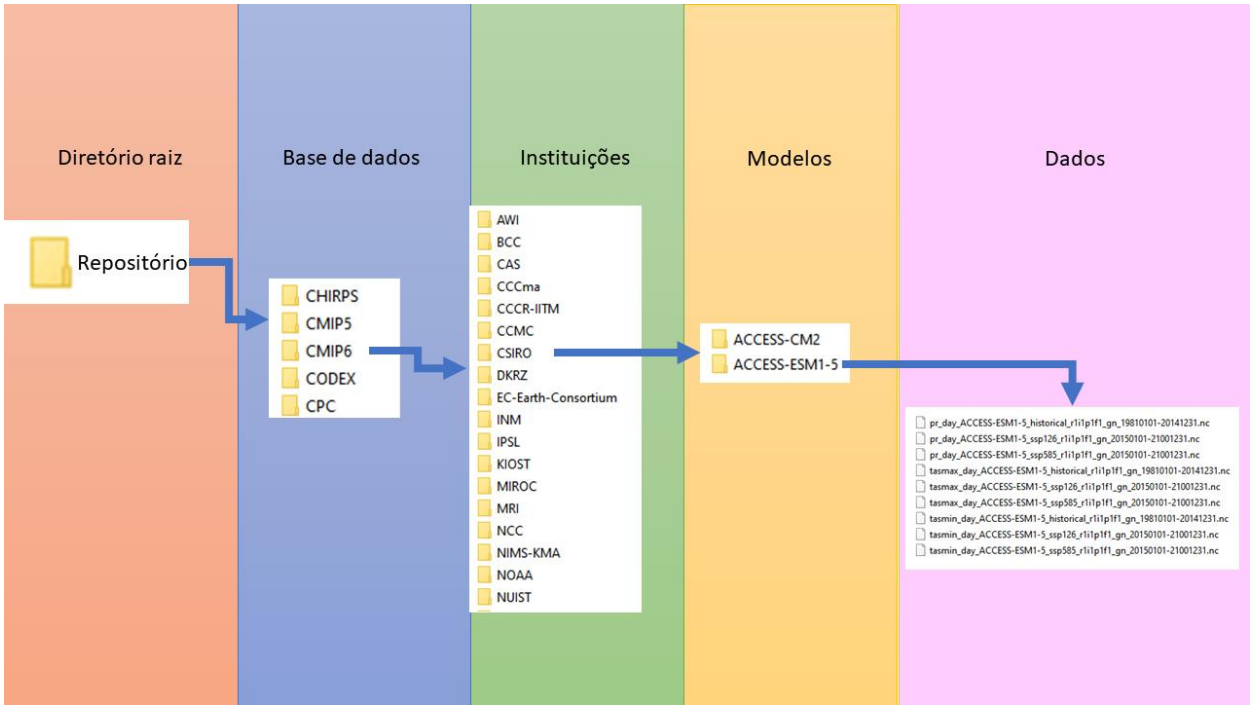

Figura 3. Informações sobre a estrutura de organização do repositório

O CMIP 6 foram separados e dentro de cada diretório encontramos uma divisão por instituições (descritas na tabela 2) e em cada diretório das instituições, temos um diretório de cada modelo dessa instituição. Seguindo o exemplo de navegação da figura 3, temos acessamos o repositório, depois CMIP6, passamos pela instituição CSIRO e encontramos 2 diretórios dos modelos ACCESS-CM2 e ACCES-ESM1-5, acessando o diretório do modelo ACCES-ESM1-5, já encontramos os dados.

A nomenclatura dos arquivos também faz parte da estrutura e organização dos dados e o significado dessa nomenclatura é descrita na figura 4.

Com o intuito de facilitar na distribuição de diretórios, a nomenclatura trás informações importantes sobre o arquivo em questão, evitando um número excessivo de diretórios e trazendo recursividade a informação importante que é o nome do modelo.

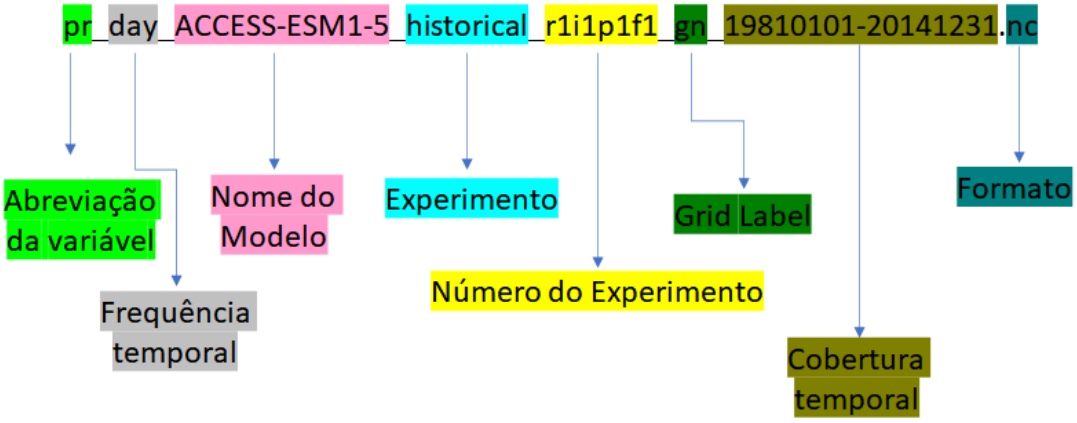

Figura 4. Descrição da nomenclatura dos dados do CMIP6.

A Tabela 1 descreve as características dos dados contidos no repositório. Tem-se a informação da resolução espacial de cada base de dados, o período, variáveis (nomenclatura), quantidade de registros, tamanho.

Tabela 1. Informações sobre o repositório.

Dados	Resolução	Cobertura Temporal	Variáveis	Tamanho
CPC	0,5º	1979-2020	TMAX TMIN	19,6GB
CHIRPS	0,05º	1981-2020	PREC	88,8GB
CORDEX	SAM 20,22,44	1960-2100	TMAX, TMIN e PREC	1,56TB
CMIP6	1º	1850-2100	TMAX, TMIN e PREC	600GB

Como é possível observar na tabela 2, para esse repositório foram selecionados 30 modelos climáticos do CMIP6. Vale salientar que para o CMIP6, os dados foram duplicados, eles estão padronizados na resolução descrita na tabela 1 de 1 º, porem, também foram mantidos os arquivos na resolução original de cada modelo. Os detalhes da resolução, as instituições e a referencia bibliográfica de cada modelo do CMIP6, estão presentes na tabela 2.

Tabela 2. Lista de modelos de cada experimento

Modelos	Instituição	Referencia bibliográfica	Resolução espacial
CCSM4	NCAR	https://www.cesm.ucar.edu/models/ccsm4.0/ccsm_doc/ug.pdf	250km
MPI-ESM-MR	DKRZ	https://doi.org/10.22033/ESGF/CMIP6.1621	250km
CanESM5	CCCma	https://doi.org/10.22033/ESGF/CMIP6.1317	500km
ACCESS-ESM1-5	CSIRO	https://doi.org/10.22033/ESGF/CMIP6.2288	250km
ACCESS-CM2	CSIRO	https://doi.org/10.22033/ESGF/CMIP6.4319	250km
MPI-ESM1-2-HR	DKRZ	https://doi.org/10.22033/ESGF/CMIP6.762	250km
MPI-ESM1-2-LR	DKRZ	https://doi.org/10.22033/ESGF/CMIP6.6693	250km
MPI-ESM-1-2-HAM	DKRZ	https://doi.org/10.22033/ESGF/CMIP6.1621	250km
INM-CM5-0	INM	https://doi.org/10.22033/ESGF/CMIP6.5081	100km
INM-CM4-8	INM	https://doi.org/10.22033/ESGF/CMIP6.5080	100km
IPSL-CM6A-LR	IPSL	https://doi.org/10.22033/ESGF/CMIP6.1534	250km
MIROC6	MIROC	https://doi.org/10.22033/ESGF/CMIP6.898	250km
MRI-ESM2-0	MRI	https://doi.org/10.22033/ESGF/CMIP6.621	100km
GFDL-ESM4	NOAA	https://doi.org/10.22033/ESGF/CMIP6.1404	100km
NESM3	NUIST	https://doi.org/10.22033/ESGF/CMIP6.2021	250km
AWI-ESM-1-1-LR	AWI	https://doi.org/10.22033/ESGF/CMIP6.9328	250km
EC-Earth3-Veg-LR	EC-Earth-Consortium	https://doi.org/10.22033/ESGF/CMIP6.718	250km
EC-Earth3	EC-Earth-Consortium	https://doi.org/10.22033/ESGF/CMIP6.4700	100km
EC-Earth3-CC	EC-Earth-Consortium	https://doi.org/10.22033/ESGF/CMIP6.4700	100km
EC-Earth3-Veg	EC-Earth-Consortium	https://doi.org/10.22033/ESGF/CMIP6.4524	100km
FGOALS-f3-L	CAS	https://doi.org/10.22033/ESGF/CMIP6.3054	250km
KIOST-ESM	KIOST	https://doi.org/10.22033/ESGF/CMIP6.1922	250km
NorESM2-LM	NCC	https://doi.org/10.22033/ESGF/CMIP6.502	250km
NorESM2-MM	NCC	https://doi.org/10.22033/ESGF/CMIP6.506	100km
BCC-CSM2-MR	BCC	https://doi.org/10.22033/ESGF/CMIP6.1725	100km
CMCC-CM2-SR5	CMCC	https://doi.org/10.22033/ESGF/CMIP6.1358	100km
CMCC-CM2-HR4	CMCC	https://doi.org/10.22033/ESGF/CMIP6.1362	100km
CMCC-ESM2	CMCC	https://doi.org/10.22033/ESGF/CMIP6.13165	100km
IITM-ESM	CCCR-IITM	https://doi.org/10.22033/ESGF/CMIP6.44	250km
KACE-1-0-G	NIMS-KMA	https://doi.org/10.22033/ESGF/CMIP6.2241	250km

Na tabela 3, podemos obter informações referentes as forçantes utilizadas no modelo CORDEX, as forçantes (FORÇANTES CMIP5 na tabela 3) são dados de simulações do CMIP5 que foram utilizadas como condições iniciais nos modelos regionais. Esses modelos regionais estão informados na coluna MODELO da tabela 3. O domínio listado como SAM (South AMerica) é o mesmo da figura 2, e os números 20, 22 e 44 que compõem a sigla, significam a resolução em quilómetros que cada modelo regional foi simulado ou seja, 20km, 22km e 44km.

Tabela 3 – Lista de modelos utilizados como forçantes no projeto CORDEX

DOMÍNIO	INSTITUTO	MODELO	FORÇANTES (CMIP5)	VOLUME
SAM 20	INPE	ETA	HadGEM2-ES, MIROC5	261 GB
SAM 22	GERICS	REMO	HadGEM2-ES, NorESM1-M MPI-M-MPI-ESM-MR	204 GB
	ICTP	ReGCM4.7	HadGEM2-ES, NorESM1-M MPI-M-MPI-ESM-MR	102 GB

SAM 44	UCAN	WRF3.4	CCCma-CanESM2	10.8 GB
	ICTP	ReGCM4.3	HadGEM2-ES, MPI-M-MPI-ESM-MR	32.4 GB
			NOAA-GFDL-GFDL-ESM2M	
	MPI-CSC	REMO2009.v1	MPI-M-MPI-ESM-LR	10.7 GB
	SMHI	RCA4	CCCma-CanESM2, CSIRO-Mk3-6-0	97.06 GB
			EC-EARTH, IPSL-CM5A-MR	
			MIROC5, HadGEM2-ES	
			MPI-M-MPI-ESM-LR, NorESM1-M	

5. Considerações finais

O repositório contem duas fontes de dados observados, CPC e CHIRPS que somam 110 Gigabytes e a base de projeções possui 22 modelos do projeto CORDEX e 30 modelos climáticos do CMIP6 e para cada modelo, selecionadas 3 variáveis, temperatura máxima, temperatura mínima e precipitação, todas na escala temporal diária e recorte da área de domínio América do Sul. Combinados com os diferentes cenários de emissões de GEE (baixas e altas emissões), totalizam um conjunto de 90 projeções de clima na escala global e 66 na escala regional, ambos, para um horizonte temporal até 2100. O repositório tem no total um tamanho de 2,8 Terabytes.

Os dados estão organizados, padronizados e disponíveis para serem utilizado na construção de análises de riscos climáticos e atender as necessidades das iniciativas do Ministério de Infraestrutura, Movimento Viva Água, Defesa Civil de Santa Cataria e Agência Nacional de Transportes Aquaviários.

Sua principal limitação é manter os dados atualizados, completar os dados à partir de outras fontes de dados, e corrigir novas inconsistências que porventura forem detectadas.

6. Referências

REMEDIO, A. R. et al. Evaluation of new cordex simulations using an updated köppen–trewartha climate classification. Atmosphere, Multidisciplinary Digital Publishing Institute, v. 10, n. 11, p. 726, 2019.

ANEXO 1

SCRIPT DE DOWNLOAD DOS ARQUIVOS DO MODELO BCC-CSM2-MR PARA O EXPERIMENTO HISTORICO

```
#!/bin/bash
##############################################################################
##
# ESG Federation download script
#
# Template version: 1.2
# Generated by esgf-node.llnl.gov - 2021/01/27 06:08:04
# Search URL: https://esgf-node.llnl.gov/esg-
search/wget/?distrib=false&dataset_id=CMIP6.CMIP.BCC.BCC-CSM2-
MR.historical.r1i1p1f1.day.tasmax.gn.v20201019|cmip.bcc.cma.cn
#
##############################################################################
###
# first be sure it's bash... anything out of bash or sh will break
# and the test will assure we are not using sh instead of bash
if [ $BASH ] && [ `basename $BASH` != bash ]; then
   echo "######## This is a bash script! ###############"
   echo "Change the execution bit 'chmod u+x $0' or start with 'bash $0' instead of sh."
   echo "Trying to recover automatically..."
   sleep 1
   /bin/bash $0 $@
   exit $?
fi

version=1.3.2
CACHE_FILE=.$(basename $0).status
openId=
search_url='https://esgf-node.llnl.gov/esg-
search/wget/?distrib=false&dataset_id=CMIP6.CMIP.BCC.BCC-CSM2-
MR.historical.r1i1p1f1.day.tasmax.gn.v20201019|cmip.bcc.cma.cn'

#These are the embedded files to be downloaded
download_files="$(cat <<EOF--dataset.file.url.chksum_type.chksum
'tasmax_day_BCC-CSM2-MR_historical_r1i1p1f1_gn_18500101-18741231.nc'
'http://cmip.bcc.cma.cn/thredds/fileServer/cmip6_data/CMIP/BCC/BCC-CSM2-
MR/historical/r1i1p1f1/day/tasmax/gn/v20201019/tasmax_day_BCC-CSM2-
MR_historical_r1i1p1f1_gn_18500101-18741231.nc' 'SHA256'
'e2ce157e0b5b69311314122fdb5e677cccd5b2d5bca0ddb5dd957fd7d8ca2797'
'tasmax_day_BCC-CSM2-MR_historical_r1i1p1f1_gn_18750101-18991231.nc'
'http://cmip.bcc.cma.cn/thredds/fileServer/cmip6_data/CMIP/BCC/BCC-CSM2-
MR/historical/r1i1p1f1/day/tasmax/gn/v20201019/tasmax_day_BCC-CSM2-
MR_historical_r1i1p1f1_gn_18750101-18991231.nc' 'SHA256'
'3f21314fee8b523ab93457353264d4a7c438322bace9e9733cd92d7f93ea7be2'
'tasmax_day_BCC-CSM2-MR_historical_r1i1p1f1_gn_19000101-19241231.nc'
'http://cmip.bcc.cma.cn/thredds/fileServer/cmip6_data/CMIP/BCC/BCC-CSM2-
MR/historical/r1i1p1f1/day/tasmax/gn/v20201019/tasmax_day_BCC-CSM2-
MR_historical_r1i1p1f1_gn_19000101-19241231.nc' 'SHA256'
'73327976205251da6b767be2a5de3c983381fd4ab3a828a7fb2aefb534bf2f70'
'tasmax_day_BCC-CSM2-MR_historical_r1i1p1f1_gn_19250101-19491231.nc'
'http://cmip.bcc.cma.cn/thredds/fileServer/cmip6_data/CMIP/BCC/BCC-CSM2-
```

```
MR/historical/r1i1p1f1/day/tasmax/gn/v20201019/tasmax_day_BCC-CSM2-
MR_historical_r1i1p1f1_gn_19250101-19491231.nc' 'SHA256'
'787372bb1fe9a082163bda54b7a8641ac5bc498a6321cf425ce25d15c1cc2d48'
'tasmax_day_BCC-CSM2-MR_historical_r1i1p1f1_gn_19500101-19741231.nc'
'http://cmip.bcc.cma.cn/thredds/fileServer/cmip6_data/CMIP/BCC/BCC-CSM2-
MR/historical/r1i1p1f1/day/tasmax/gn/v20201019/tasmax_day_BCC-CSM2-
MR_historical_r1i1p1f1_gn_19500101-19741231.nc' 'SHA256'
'67561da07865b7e5921469424a88edc4c927fd7c46a01d04f84b9d5f677e936d'
'tasmax_day_BCC-CSM2-MR_historical_r1i1p1f1_gn_19750101-19991231.nc'
'http://cmip.bcc.cma.cn/thredds/fileServer/cmip6_data/CMIP/BCC/BCC-CSM2-
MR/historical/r1i1p1f1/day/tasmax/gn/v20201019/tasmax_day_BCC-CSM2-
MR_historical_r1i1p1f1_gn_19750101-19991231.nc' 'SHA256'
'30b8f7dbf6ad516edf625979f8575aaac98045b92afe634c1add7033dab326e4'
'tasmax_day_BCC-CSM2-MR_historical_r1i1p1f1_gn_20000101-20141231.nc'
'http://cmip.bcc.cma.cn/thredds/fileServer/cmip6_data/CMIP/BCC/BCC-CSM2-
MR/historical/r1i1p1f1/day/tasmax/gn/v20201019/tasmax_day_BCC-CSM2-
MR_historical_r1i1p1f1_gn_20000101-20141231.nc' 'SHA256'
'47039a30cd732e54c54ee092f81254916ece7b9b79b91d992cb800b3d194d9e1'
EOF--dataset.file.url.chksum_type.chksum
)"

# ESG_HOME should point to the directory containing ESG credentials.
#   Default is $HOME/.esg
ESG_HOME=${ESG_HOME:-$HOME/.esg}
[[ -d $ESG_HOME ]] || mkdir -p $ESG_HOME

ESG_CREDENTIALS=${X509_USER_PROXY:-$ESG_HOME/credentials.pem}
ESG_CERT_DIR=${X509_CERT_DIR:-$ESG_HOME/certificates}
MYPROXY_STATUS=$HOME/.MyProxyLogon
COOKIE_JAR=$ESG_HOME/cookies
MYPROXY_GETCERT=$ESG_HOME/getcert.jar
CERT_EXPIRATION_WARNING=$((60 * 60 * 8))   #Eight hour (in seconds)

WGET_TRUSTED_CERTIFICATES=$ESG_HOME/certificates


# Configure checking of server SSL certificates.
#   Disabling server certificate checking can resolve problems with myproxy
#   servers being out of sync with datanodes.
CHECK_SERVER_CERT=${CHECK_SERVER_CERT:-Yes}

check_os() {
   local os_name=$(uname | awk '{print $1}')
   case ${os_name} in
      Linux)
         ((debug)) && echo "Linux operating system detected"
         LINUX=1
         MACOSX=0
         ;;
      Darwin)
         ((debug)) && echo "Mac OS X operating system detected"
         LINUX=0
         MACOSX=1
         ;;
      *)
         echo "Unrecognized OS [${os_name}]"
         return 1
```

```
            ;;
    esac
    return 0
}

#taken from http://stackoverflow.com/a/4025065/1182464
vercomp () {
    if [[ $1 == $2 ]]
    then
        return 0
    fi
    local IFS=.
    local i ver1=($1) ver2=($2)
    # fill empty fields in ver1 with zeros
    for ((i=${#ver1[@]}; i<${#ver2[@]}; i++))
    do
        ver1[i]=0
    done
    for ((i=0; i<${#ver1[@]}; i++))
    do
        if [[ -z ${ver2[i]} ]]
        then
            # fill empty fields in ver2 with zeros
            ver2[i]=0
        fi
        if ((10#${ver1[i]} > 10#${ver2[i]}))
        then
            return 1
        fi
        if ((10#${ver1[i]} < 10#${ver2[i]}))
        then
            return 2
        fi
    done
    return 0
}

check_commands() {
    #check wget
    local MIN_WGET_VERSION=1.10
    vercomp $(wget -V | sed -n 's/^.* \([1-9]\.[0-9.]*\) .*$/\1/p') $MIN_WGET_VERSION
    case $? in
        2) #lower
            wget -V
            echo
            echo "** ERROR: wget version is too old. Use version $MIN_WGET_VERSION or
greater. **" >&2
            exit 1
    esac
}

usage() {
    echo "Usage: $(basename $0) [flags] [openid] [username]"
    echo "Flags is one of:"
    sed -n '/^while getopts/,/^done/  s/^\([^)]*\)[^#]*#\(.*$\)/\1 \2/p' $0
    echo
    echo "This command stores the states of the downloads in .$0.status"
```

```
    echo "For more information check the website: http://esgf.org/wiki/ESGF_wget"
}

#defaults
debug=0
clean_work=1

#parse flags
while getopts ':c:pfF:o:w:isuUndvqhHI:T' OPT; do
    case $OPT in
        H) skip_security=1 && use_http_sec=1;; #       : Authenticate with OpenID (username,) and password, without the need for a certificate.
        T) force_TLSv1=1;;             #       : Forces wget to use TLSv1.
        c) ESG_CREDENTIALS="$OPTARG";;  #<cert> : use this certificate for authentication.
        f) force=1;;                   #       : force certificate retrieval (defaults to only once per day); for certificate-less authentication (see -H option), this flag will force login and refresh cookies.
        F) input_file="$OPTARG";;       #<file> : read input from file instead of the embedded one (use - to read from stdin)
        o) openId="$OPTARG";;          #<openid>: Provide OpenID instead of interactively asking for it.
        I) username_supplied="$OPTARG";;    #<user_id> : Explicitly set user ID.  By default, the user ID is extracted from the last component of the OpenID URL.  Use this flag to override this behaviour.
        w) output="$OPTARG";;          #<file> : Write embedded files into a file and exit
        i) insecure=1;;                #       : set insecure mode, i.e. don't check server certificate
        s) skip_security=1 && use_cookies_for_http_basic_auth_start=1;;            #       : completely skip security. It will only work if the accessed data is not secured at all. -- works only if the accessed data is unsecured or a certificate exists or cookies are saved (latter applies to -H option only).
        u) update=1;;                  #       : Issue the search again and see if something has changed.
        U) update_files=1;;            #       : Update files from server overwriting local ones (detect with -u)
        n) dry_run=1;;                 #       : Don't download any files, just report.
        p) clean_work=0;;              #       : preserve data that failed checksum
        d) verbose=1;debug=1;;         #       : display debug information
        v) verbose=1;;                 #       : be more verbose
        q) quiet=1;;                   #       : be less verbose
        h) usage && exit 0;;           #       : displays this help
        \?) echo "Unknown option '$OPTARG'" >&2 && usage && exit 1;;
        \:) echo "Missing parameter for flag '$OPTARG'" >&2 && usage && exit 1;;
    esac
done
shift $(($OPTIND - 1))

#setup input as desired by the user
if [[ "$input_file" ]]; then
    if [[ "$input_file" == '-' ]]; then
        download_files="$(cat)" #read from STDIN
        exec 0</dev/tty #reopen STDIN as cat closed it
    else
        download_files="$(cat $input_file)" #read from file
    fi
fi

#if -w (output) was selected write file and finish:
if [[ "$output" ]]; then
    #check the file
```

```
    if [[ -f "$output" ]]; then
        read -p "Overwrite existing file $output? (y/N) " answ
        case $answ in y|Y|yes|Yes);; *) echo "Aborting then..."; exit 0;; esac
    fi
    echo "$download_files">$output
    exit
fi


#assure we have everything we need
check_commands

if ((update)); then
    echo "Checking the server for changes..."
    new_wget="$(wget "$search_url" -qO -)"
    compare_cmd="grep -vE '^(# Generated by|# Search URL|search_url=)'"
    if diff -q <(eval $compare_cmd<<<"$new_wget") <(eval $compare_cmd $0) >/dev/null; then
        echo "No changes detected."
    else
        echo "Wget was changed. Dowloading. (old renamed to $0.old.#N)"
        counter=0
        while [[ -f $0.old.$counter ]]; do ((counter++)); done
        mv $0 $0.old.$counter
        echo "$new_wget" > $0
    fi
    exit 0
fi


################################################################################
##
check_java() {
    if ! type java >& /dev/null; then
        echo "Java could not be found." >&2
        return 1
    fi
    if java -version 2>&1|grep openjdk >/dev/null; then
        openjdk=1;
    else
        openjdk=0;
    fi
    jversion=($(jversion=$(java -version 2>&1 | awk '/version/ {gsub("\"","");print $3}'); echo
${jversion//./ }))
    mVer=${jversion[1]}
    if [ $openjdk -eq 1 ]; then
        mVer=${jversion[0]}
        if ((mVer<5)); then
            echo "Openjdk detected. Version 9+ is required for retrieving the certificate." >&2
            echo "Current version seems older: $(java -version | head -n1) " >&2
            return 1
        fi
    else

        if ((mVer<5)); then
            echo "Java version 1.5+ is required for retrieving the certificate." >&2
            echo "Current version seems older: $(java -version | head -n1) " >&2
            return 1
```

```
        fi
    fi
}

check_myproxy_logon() {
    if ! type myproxy-logon >& /dev/null; then
            echo "myproxy-logon could not be found." >&2
            return 1
    fi
    echo "myproxy-logon found" >&2
}

proxy_to_java() {
    local proxy_user proxy_pass proxy_server proxy_port
    eval $(sed 's#^\(https\?://\)\?\(\([^:@]*\)\(:\([^@]*\)\)\?@\)\?\([^:/]*\)\(:\([0-
9]*\)\)\?.*#proxy_user=\3;proxy_pass=\5;proxy_server=\6;proxy_port=\8#'<<<$http_proxy)
    local JAVA_PROXY=
    [[ "$proxy_server" ]] && JAVA_PROXY=$JAVA_PROXY" -Dhttp.proxyHost=$proxy_server"
    [[ "$proxy_port" ]] && JAVA_PROXY=$JAVA_PROXY" -Dhttp.proxyPort=$proxy_port"
    eval $(sed 's#^\(https\?://\)\?\(\([^:@]*\)\(:\([^@]*\)\)\?@\)\?\([^:/]*\)\(:\([0-
9]*\)\)\?.*#proxy_user=\3;proxy_pass=\5;proxy_server=\6;proxy_port=\8#'<<<$https_proxy)
    [[ "$proxy_server" ]] && JAVA_PROXY=$JAVA_PROXY" -Dhttps.proxyHost=$proxy_server"
    [[ "$proxy_port" ]] && JAVA_PROXY=$JAVA_PROXY" -Dhttps.proxyPort=$proxy_port"

    echo "$JAVA_PROXY"
}

# get certificates from github
get_certificates() {
    # don't if this was already done today
    [[ -z $force && "$(find $ESG_CERT_DIR -type d -mtime -1 2>/dev/null)" ]] && return 0
    echo -n "Retrieving Federation Certificates..." >&2

    if ! wget -O $ESG_HOME/esg-truststore.ts --no-check-certificate
https://github.com/ESGF/esgf-dist/raw/master/installer/certs/esg-truststore.ts; then
        echo "Could not fetch esg-truststore";
        return 1
    fi

    if ! wget --no-check-certificate https://raw.githubusercontent.com/ESGF/esgf-
dist/master/installer/certs/esg_trusted_certificates.tar -O - -q | tar x -C $ESG_HOME; then
        #certificates tarred into esg_trusted_certificates. (if it breaks, let the user know why
        wget --no-check-certificate https://raw.githubusercontent.com/ESGF/esgf-
dist/master/installer/certs/esg_trusted_certificates.tar
        echo "Could't update certs!" >&2
        return 1
    else
        #if here everythng went fine. Replace old cert with this ones
        [[ -d $ESG_CERT_DIR ]] && rm -r $ESG_CERT_DIR || mkdir -p $(dirname
$ESG_CERT_DIR)
        mv $ESG_HOME/esg_trusted_certificates $ESG_CERT_DIR
        touch $ESG_CERT_DIR
        echo "done!" >&2
    fi

}
```

```
# Retrieve ESG credentials
unset pass
get_credentials() {
   if check_java
   then
         use_java=1
   else
         use_java=0
         echo "No suitable java for obtaining certificate - checking for myproxy-logon instead"
>&2
         check_myproxy_logon || exit 1
   fi
   #get all certificates
   get_certificates

   if [[ -z "$(find $MYPROXY_GETCERT -type f -mtime -1 2>/dev/null)" ]]; then
      echo -n "(Downloading $MYPROXY_GETCERT... "
      mkdir -p $(dirname $MYPROXY_GETCERT)
      if wget -q --no-check-certificate https://raw.githubusercontent.com/ESGF/esgf-
dist/master/installer/certs/getcert.jar -O $MYPROXY_GETCERT;then
         echo 'done)'
         touch $MYPROXY_GETCERT
      else
         echo 'failed)'
      fi
   fi

   #if the user already defined one, use it
   if [[ -z $openId ]]; then
      #try to parse the last valid value if any
      [[ -f "$MYPROXY_STATUS" ]] && openId=$(awk -F= '/^OpenID/ {gsub("\\\\", ""); print $2}'
$MYPROXY_STATUS)
      if [[ -z $openId ]]; then
         #no OpenID, we need to ask the user
         echo -n "Please give your OpenID (Example: https://myserver/example/username) ? "
      else
         #Allow the user to change it if desired
         echo -n "Please give your OpenID (hit ENTER to accept default: $openId)? "
      fi
      read -e
      [[ "$REPLY" ]] && openId="$REPLY"
   else
      ((verbose)) && echo "Using user defined OpenID $openId (to change use -o <open_id>)"
   fi

   if grep -q ceda.ac.uk <<<$openId; then
      username=${openId##*/}
      echo -n "Please give your username if different [$username]: "
      read -e
      [[ "$REPLY" ]] && username="$REPLY"
   fi



   if [ $use_java -eq 1 ]
   then
      local args=
```

```
        #get password
            [[ ! "$pass" ]] && read -sp "MyProxy Password? " pass
        [[ "$openId" ]] && args=$args" --oid $openId"
        [[ "$pass" ]] && args=$args" -P $pass"
        [[ "$username" ]] && args=$args" -l $username"

        echo -n $'\nRetrieving Credentials...' >&2
        if ! java $(proxy_to_java) -jar $MYPROXY_GETCERT $args --ca-directory
$ESG_CERT_DIR --output $ESG_CREDENTIALS ; then
            echo "Certificate could not be retrieved"
            exit 1
        fi
        echo "done!" >&2
    else
        args=`openid_to_myproxy_args $openId $username` || exit 1
        if ! myproxy-logon $args -b -o $ESG_CREDENTIALS
            then
            echo "Certificate could not be retrieved"
                exit 1
        fi
            cp $HOME/.globus/certificates/* $ESG_CERT_DIR/
    fi
}

openid_to_myproxy_args() {
  python - <<EOF || exit 1
import sys
import re
import xml.etree.ElementTree as ET
import urllib2
openid = "$1"
username = "$2" or re.sub(".*/", "", openid)
e = ET.parse(urllib2.urlopen(openid))
servs = [el for el in e.getiterator() if el.tag.endswith("Service")]
for serv in servs:
    servinfo = dict([(re.sub(".*}", "", c.tag), c.text)
                     for c in serv.getchildren()])
    try:
        if servinfo["Type"].endswith("myproxy-service"):
            m = re.match("socket://(.*):(.*)", servinfo["URI"])
            if m:
                host = m.group(1)
                port = m.group(2)
                print "-s %s -p %s -l %s" % (host, port, username)
                break
    except KeyError:
        continue
else:
    sys.stderr.write("myproxy service could not be found\n")
    sys.exit(1)
EOF
}

# check the certificate validity
check_cert() {
    if [[ ! -f "$ESG_CERT" || $force ]]; then
        #not there, just get it
```

```
        get_credentials
    elif which openssl &>/dev/null; then
        #check openssl and certificate
        if ! openssl x509 -checkend $CERT_EXPIRATION_WARNING -noout -in $ESG_CERT
2>/dev/null; then
            echo "The certificate expires in less than $((CERT_EXPIRATION_WARNING / 60 / 60))
hour(s). Renewing..."
            get_credentials
        else
            #ok, certificate is fine
            return 0
        fi
    fi
}

#
# Detect ESG credentials
#
find_credentials() {

    #is X509_USER_PROXY or $HOME/.esg/credential.pem
    if [[ -f "$ESG_CREDENTIALS" ]]; then
        # file found, proceed.
        ESG_CERT="$ESG_CREDENTIALS"
        ESG_KEY="$ESG_CREDENTIALS"
    elif [[ -f "$X509_USER_CERT" && -f "$X509_USER_KEY" ]]; then
        # second try, use these certificates.
        ESG_CERT="$X509_USER_CERT"
        ESG_KEY="$X509_USER_KEY"
    else
        # If credentials are not present, just point to where they should go
        echo "No ESG Credentials found in $ESG_CREDENTIALS" >&2
            ESG_CERT="$ESG_CREDENTIALS"
            ESG_KEY="$ESG_CREDENTIALS"
            #they will be retrieved later one
    fi


    #chek openssl and certificate
    if (which openssl &>/dev/null); then
        if ( openssl version | grep 'OpenSSL 1\.0' ); then
            echo '** WARNING: ESGF Host certificate checking might not be compatible with
OpenSSL 1.0+'
        fi
        check_cert || { (($?==1)); exit 1; }
    fi

    if [[ $CHECK_SERVER_CERT == "Yes" ]]; then
        [[ -d "$ESG_CERT_DIR" ]] || { echo "CA certs not found. Aborting."; exit 1; }
        PKI_WGET_OPTS="--ca-directory=$ESG_CERT_DIR"
    fi

    #some wget version complain if there's no file present
    [[ -f $COOKIE_JAR ]] || touch $COOKIE_JAR

    PKI_WGET_OPTS="$PKI_WGET_OPTS --certificate=$ESG_CERT --private-
key=$ESG_KEY --save-cookies=$COOKIE_JAR --load-cookies=$COOKIE_JAR --ca-
```

```
certificate=$ESG_CERT"

}

check_chksum() {
    local file="$1"
    local chk_type=$2
    local chk_value=$3
    local local_chksum=Unknown

    case $chk_type in
        md5) local_chksum=$(md5sum_ $file | cut -f1 -d" ");;
        sha256) local_chksum=$(sha256sum_ $file|awk '{print $1}'|cut -d ' ' -f1);;
        *) echo "Can't verify checksum." && return 0;;
    esac

    #verify
    ((debug)) && echo "local:$local_chksum vs remote:$chk_value" >&2
    echo $local_chksum
}

#Our own md5sum function call that takes into account machines that don't have md5sum but
do have md5 (i.e. mac os x)
md5sum_() {
    hash -r
    if type md5sum >& /dev/null; then
        echo $(md5sum $@)
    else
        echo $(md5 $@ | sed -n 's/MD5[ ]*\(.*\)[^=]*=[ ]*\(.*$\)/\2 \1/p')
    fi
}

#Our own sha256sum function call that takes into account machines that don't have
sha256sum but do have sha2 (i.e. mac os x)
sha256sum_() {
    hash -r
    if type sha256sum >& /dev/null; then
        echo $(sha256sum $@)
    elif type shasum >& /dev/null; then
        echo $(shasum -a 256 $@)
    else
        echo $(sha2 -q -256 $@)
    fi
}

get_mod_time_() {
    if ((MACOSX)); then
        #on a mac modtime is stat -f %m <file>
        echo "$(stat -f %m $@)"
    else
        #on linux (cygwin) modtime is stat -c %Y <file>
        echo "$(stat -c %Y $@)"
    fi
    return 0;
}

remove_from_cache() {
```

```
    local entry="$1"
    local tmp_file="$(grep -ve "^$entry" "$CACHE_FILE")"
    echo "$tmp_file" > "$CACHE_FILE"
    unset cached
}

#Download data from node using cookies and not certificates.
download_http_sec()
{
  #The data to be downloaded.
  data=" $url"
  filename="$file"

  #Wget args.
  if ((insecure))
  then
   wget_args=" --no-check-certificate --cookies=on  --keep-session-cookies --save-cookies
$COOKIES_FOLDER/wcookies.txt "
  else
   wget_args=" --ca-directory=$WGET_TRUSTED_CERTIFICATES --cookies=on --keep-
session-cookies --save-cookies $COOKIES_FOLDER/wcookies.txt "
  fi

  if ((use_cookies_for_http_basic_auth_start)) || ((use_cookies_for_http_basic_auth))
  then
   wget_args=" $wget_args"" --load-cookies $COOKIES_FOLDER/wcookies.txt"
  fi

  if((force_TLSv1))
  then
   wget_args=" $wget_args"" --secure-protocol=TLSv1 "
  fi


  if [[ ! -z "$ESGF_WGET_OPTS" ]]
  then
    wget_args="$wget_args $ESGF_WGET_OPTS"
  fi


  #use cookies for the next downloads
  use_cookies_for_http_basic_auth=1;

  #Debug message.
  if  ((debug))
  then
   echo -e "\nExecuting:\n"
   echo -e "wget $wget_args $data\n"
  fi


  #Try to download the data.
  command="wget $wget_args -O $filename $data"
  http_resp=$(eval $command  2>&1)
  cmd_exit_status="$?"

  if ((debug))
```

```
  then
   echo -e "\nHTTP response:\n $http_resp\n"
  fi

  #Extract orp service from url ?
  #Evaluate response.
  #redirects=$(echo "$http_resp" | egrep -c ' 302 ')
  #(( "$redirects" == 1 )) &&
  if  echo "$http_resp" | grep -q "/esg-orp/"
  then
   urls=$(echo "$http_resp" | egrep -o 'https://[^ ]+' | cut -d'/' -f 3)
   orp_service=$(echo "$urls" | tr '\n' ' ' | cut -d' ' -f 2)


   #Use cookies for transaction with orp.
   wget_args=" $wget_args"" --load-cookies $COOKIES_FOLDER/wcookies.txt"

   #Download data using either http basic auth or http login form.
   if [[ "$openid_c" == */openid/  || "$openid_c" == */openid ]]
   then
    download_http_sec_open_id
   else
    download_http_sec_decide_service
   fi
  else
   if    echo "$http_resp" | grep -q "401 Unauthorized"  \
      || echo "$http_resp" | grep -q "403: Forbidden"  \
      || echo "$http_resp" | grep -q "Connection timed out."  \
      || echo "$http_resp" | grep -q "no-check-certificate"  \
      || (( $cmd_exit_status != 0 ))
   then
    echo "ERROR : http request to OpenID Relying Party service failed."
    failed=1
   fi
  fi
}


#Function that decides which implementaion of idp to use.
download_http_sec_decide_service()
{
  #find claimed id

  pos=$(echo "$openid_c" | egrep -o '/' | wc -l)
  username_c=$(echo "$openid_c"  | cut -d'/' -f "$(($pos + 1))")
  esgf_uri=$(echo "$openid_c" | egrep -o '/esgf-idp/openid/')

  host=$(echo "$openid_c"  | cut -d'/' -f 3)
  #test ceda first.

  if [[ -z "$esgf_uri" ]]
  then
   openid_c_tmp="https://""$host""/openid/"
  else
   openid_c_tmp="https://""$host""/esgf-idp/openid/"
  fi
```

```
  command="wget "$openid_c_tmp" --no-check-certificate ${force_TLSv1:+--secure-
protocol=TLSv1} -O-"

  if [[ ! -z "$ESGF_WGET_OPTS" ]]
  then
   command="$command $ESGF_WGET_OPTS"
  fi

  #Debug message.
  if  ((debug))
  then
   echo -e "\nExecuting:\n"
   echo -e "$command\n"
  fi


  #Execution of command.
  http_resp=$(eval $command  2>&1)
  cmd_exit_status="$?"


  if ((debug))
  then
   echo -e "\nHTTP response:\n $http_resp\n"
  fi


  if    echo "$http_resp" | grep -q "[application/xrds+xml]"  \
     && echo "$http_resp" | grep -q "200 OK"  \
     && (( cmd_exit_status == 0 ))
  then
   openid_c=$openid_c_tmp
   download_http_sec_open_id
  else
   if [[ -z "$esgf_uri" ]]
   then
    echo "ERROR : HTTP request to OpenID Relying Party service failed."
    failed=1
   else
   download_http_sec_cl_id
   fi
  fi
}


download_http_sec_retry()
{
  echo -e "\nRetrying....\n"
  #Retry in case that last redirect did not work, this happens with older version of wget.
  command="wget $wget_args $data"

  #Debug message.
  if  ((debug))
  then
   echo -e "Executing:\n"
   echo -e "$command\n"
  fi
```

```
  http_resp=$(eval $command  2>&1)
  cmd_exit_status="$?"

  if ((debug))
  then
   echo -e "\nHTTP response:\n $http_resp\n"
  fi

  if    echo "$http_resp" | grep -q "401 Unauthorized"  \
     || echo "$http_resp" | grep -q "403: Forbidden"  \
     || echo "$http_resp" | grep -q "Connection timed out."  \
     || echo "$http_resp" | grep -q "no-check-certificate"  \
     || (( $cmd_exit_status != 0 ))
  then
   echo -e "\nERROR : Retry failed.\n"
   #rm "$filename"
   failed=1
  fi #if retry failed.
}

#Function for downloading data using the claimed id.
download_http_sec_cl_id()
{
  #Http request for sending openid to the orp service.
  command="wget --post-data \"openid_identifier=$openid_c&rememberOpenid=on\"
$wget_args -O- https://$orp_service/esg-orp/j_spring_openid_security_check.htm "

  #Debug message.
  if  ((debug))
  then
   echo -e "Executing:\n"
   echo -e "wget $command\n"
  fi


  #Execution of command.
  http_resp=$(eval $command  2>&1)
  cmd_exit_status="$?"


  if ((debug))
  then
   echo -e "\nHTTP response:\n $http_resp\n"
  fi


  #Extract orp service from openid ?
  #Evaluate response.If redirected to idp service send the credentials.
  #redirects=$(echo "$http_resp" | egrep -c ' 302 ')
  #(( redirects == 2  )) &&
  if  echo "$http_resp" | grep -q "login.htm"  && (( cmd_exit_status == 0 ))
  then

   urls=$(echo "$http_resp" | egrep -o 'https://[^ ]+' | cut -d'/' -f 3)
   idp_service=$(echo "$urls"  | tr '\n' ' ' | cut -d' ' -f 2)
```

```
   command="wget --post-data  password=\"$password_c\" $wget_args ${quiet:+-q} ${quiet:--v}
-O $filename https://$idp_service/esgf-idp/idp/login.htm"


   #Debug message.
   if  ((debug))
   then
    echo -e "Executing:\n"
    echo -e "wget $command\n"
   fi

   #Execution of command.
   http_resp=$(eval $command  2>&1)
   cmd_exit_status="$?"

   if ((debug))
   then
    echo -e "\nHTTP response:\n $http_resp\n"
   fi

   #Evaluate response.
   #redirects=$(echo "$http_resp" | egrep -c ' 302 ')
   #(( "$redirects" != 5 )) \
   if    echo "$http_resp" | grep -q "text/html"  \
      || echo "$http_resp" | grep -q "403: Forbidden"  \
      || (( cmd_exit_status != 0 ))
   then
    rm "$filename"
    download_http_sec_retry
   fi

  else
   echo "ERROR : HTTP request to OpenID Provider service failed."
   failed=1
  fi #if redirected to idp.
}



download_http_sec_open_id()
{
  #Http request for sending openid to the orp web service.
  command="wget --post-data \"openid_identifier=$openid_c&rememberOpenid=on\" --
header=\"esgf-idea-agent-type:basic_auth\" --http-user=\"$username_c\" --http-
password=\"$password_c\"  $wget_args ${quiet:+-q} ${quiet:--v} -O $filename
https://$orp_service/esg-orp/j_spring_openid_security_check.htm "


  #Debug message.
  if  ((debug))
  then
   echo -e "Executing:\n"
   echo -e "$command\n"
  fi

  #Execution of command.
  http_resp=$(eval $command  2>&1)
```

```
  cmd_exit_status="$?"


  if ((debug))
  then
   echo -e "\nHTTP response:\n $http_resp\n"
  fi

  #Evaluate response.
  #redirects=$(echo "$http_resp" | egrep -c ' 302 ')
  #(( "$redirects" != 7 )) ||
  if   echo "$http_resp" | grep -q "text/html"  ||  (( $cmd_exit_status != 0 ))
  then
   rm "$filename"
   download_http_sec_retry
  fi #if error during http basic authentication.

}


download() {
    wget="wget ${insecure:+--no-check-certificate} ${quiet:+-q} ${quiet:--v} -c ${force_TLSv1:+--
secure-protocol=TLSv1} $PKI_WGET_OPTS"

    while read line
    do
        # read csv here document into proper variables
        eval $(awk -F "' "' '{$0=substr($0,2,length($0)-2); $3=tolower($3); print
"file=\""$1"\";url=\""$2"\";chksum_type=\""$3"\";chksum=\""$4"\""}' <(echo $line) )

        #Process the file
        echo -n "$file ..."

        #get the cached entry if any.
        cached="$(grep -e "^$file" "$CACHE_FILE")"

        #if we have the cache entry but no file, clean it.
        if [[ ! -f $file && "$cached" ]]; then
            #the file was removed, clean the cache
            remove_from_cache "$file"
            unset cached
        fi

        #check it wasn't modified
        if [[ -n "$cached" && "$(get_mod_time_ $file)" == $(echo "$cached" | cut -d ' ' -f2) ]]; then
                if [[ "$chksum" == "$(echo "$cached" | cut -d ' ' -f3)" ]]; then
            echo "Already downloaded and verified"
            continue
            elif ((update_files)); then
                #user want's to overwrite newer files
                rm $file
                remove_from_cache "$file"
                unset cached
            else
                #file on server is different from what we have.
                echo "WARNING: The remote file was changed (probably a new version is available).
Use -U to Update/overwrite"
```

```
        continue
    fi
fi
unset chksum_err_value chksum_err_count

while : ; do
    # (if we had the file size, we could check before trying to complete)
    echo "Downloading"
    [[ ! -d "$(dirname "$file")" ]] && mkdir -p "$(dirname "$file")"
    if ((dry_run)); then
        #all important info was already displayed, if in dry_run mode just abort
        #No status will be stored
        break
    else
        if ((use_http_sec))
        then
         download_http_sec
         if ((failed))
         then
          break
         fi
        else
         $wget -O "$file" $url || { failed=1; break; }
        fi
    fi

    #check if file is there
    if [[ -f $file ]]; then
        ((debug)) && echo file found
        if [[ ! "$chksum" ]]; then
            echo "Checksum not provided, can't verify file integrity"
            break
        fi
        result_chksum=$(check_chksum "$file" $chksum_type $chksum)
        if [[ "$result_chksum" != "$chksum" ]]; then
            echo "  $chksum_type failed!"
            if ((clean_work)); then
                if !((chksum_err_count)); then
                        chksum_err_value=$result_chksum
                        chksum_err_count=2
                    elif ((checksum_err_count--)); then
                        if [[ "$result_chksum" != "$chksum_err_value" ]]; then
                            #this is a real transmission problem
                            chksum_err_value=$result_chksum
                            chksum_err_count=2
                        fi
                    else
                        #ok if here we keep getting the same "different" checksum
                        echo "The file returns always a different checksum!"
                        echo "Contact the data owner to verify what is happening."
                        echo
                        sleep 1
                        break
                    fi

                    rm $file
                    #try again
```

```
                    echo -n "  re-trying..."
                    continue
                else
                    echo "  don't use -p or remove manually."
                fi
            else
                echo "  $chksum_type ok. done!"
                echo "$file" $(get_mod_time_ "$file") $chksum >> $CACHE_FILE
            fi
        fi
        #done!
        break
    done

    if ((failed)); then
        echo "download failed"
        # most common failure is certificate expiration, so check this
        #if we have the pasword we can retrigger download
        ((!skip_security)) && [[ "$pass" ]] && check_cert
        unset failed
    fi

done <<<"$download_files"

}

dedup_cache_() {
  local file=${1:-${CACHE_FILE}}
  ((debug)) && echo "dedup'ing cache ${file} ..."
  local tmp=$(LC_ALL='C' sort  -r -k1,2 $file | awk '!($1 in a) {a[$1];print $0}' | sort -k2,2)
  ((DEBUG)) && echo "$tmp"
  echo "$tmp" > $file
  ((debug)) && echo "(cache dedup'ed)"
}

http_basic_auth_func_info_message()
{
  echo  "*******************************************************************************"
  echo  "*                                                                             *"
  echo  "* Note that new functionality to allow authentication without the need for    *"
  echo  "* certificates is available with this version of the wget script.  To enable, *"
  echo  "* use the \"-H\" option and enter your OpenID and password when prompted:      *"
  echo  "*                                                                             *"
  echo  "* $ "$(basename "$0")" -H [options...]                                        *"
  echo  "*                                                                             *"
  echo  "* For a full description of the available options use the help option:       *"
  echo  "*                                                                             *"
  echo  "* $ "$(basename "$0")" -h                                                     *"
  echo  "*                                                                             *"
  echo  "*******************************************************************************"
}

#
# MAIN
#

if ((!use_http_sec))
```

```
then
 http_basic_auth_func_info_message
fi

echo "Running $(basename $0) version: $version"
((verbose)) && echo "we use other tools in here, don't try to user their proposed 'options' directly"
echo "Use $(basename $0) -h for help."$'\n'

((debug)) && cat<<EOF
** Debug info **
ESG_HOME=$ESG_HOME
ESG_CREDENTIALS=$ESG_CREDENTIALS
ESG_CERT_DIR=$ESG_CERT_DIR
** -- ** -- ** -- ** --

EOF


cat <<'EOF-MESSAGE'
Script created for 7 file(s)
(The count won't match if you manually edit this file!)



EOF-MESSAGE
sleep 1

check_os
((!skip_security)) && find_credentials

if ((use_http_sec))
then

 if (( ! insecure))
 then
  get_certificates
 fi

 #Cookies folder.
 COOKIES_FOLDER="$ESG_HOME/wget_cookies"

 if (( force ))
 then
  if [ -d $COOKIES_FOLDER ]
  then
   rm -rf $COOKIES_FOLDER
  fi
 fi

 #Create cookies folder.
 if [[ ! -d $COOKIES_FOLDER ]]
 then
  mkdir $COOKIES_FOLDER
 fi

 if((! use_cookies_for_http_basic_auth_start))
```

```
 then

  #Read openid.
  if [[ ! -z "$openId" ]]
  then
   openid_c="$openId"
  elif ( (("$#" > 1)) || (("$#" == 1)) )
  then
   openid_c=$1
  else
   read -p    "Enter your openid : " openid_c
  fi


  #Read username.
  if [[ ! -z "$username_supplied" ]]
  then
   username_c="$username_supplied"
  elif (("$#" == 2))
  then
   username_c=$2
  elif [[ "$openid_c" == */openid/ || "$openid_c" == */openid ]]
  then
   read -p    "Enter username : " username_c
  fi

  #Read password.
  read -s -p "Enter password : " password_c
  echo -e "\n"

 fi #use cookies

fi #use_http_sec


#do we have old results? Create the file if not
[ ! -f $CACHE_FILE ] && echo "#filename mtime checksum" > $CACHE_FILE && chmod 666
$CACHE_FILE

#clean the force parameter if here (at htis point we already have the certificate)
unset force

download

dedup_cache_


echo "done"
```

ANEXO 2

PADRONIZAÇÃO DO DOMINIO E RESOLUÇÃO

```
#!/bin/bash -x

mkdir box
for file in $(ls *.nc);
do

cdo -remapbil,griddes -sellonlatbox,270,330,-60,15 $file box/$file;

done

exit
```