UNIVERSIDADE ABERTA

Visualização imersiva de *big data* com dispositivos móveis

Tiago Martins Peres

Mestrado em Tecnologias e Sistemas Informáticos Web

Anexos

2020

Índice de Anexos

I. Lista de locais onde encontrar *big data*

Tabela 1. *Big Data*: conjuntos de dados gratuitos que qualquer pessoa pode utilizar.

Nome	Sítio	Último acesso
KAPSARC Data Portal	https://datasource.kapsarc.org/pages/home/	15 de abril de 2019
Kaggle	https://www.kaggle.com/datasets	15 de abril de 2019
AssetMacro	https://www.assetmacro.com/	15 de abril de 2019
Registry on Open Data on AWS	http://aws.amazon.com/datasets	15 de abril de 2019
Re3Data	https://www.re3data.org/	15 de abril de 2019
DataCite	https://datacite.org/	15 de abril de 2019
r/datasets	https://www.reddit.com/r/datasets	15 de abril de 2019
The Web Miner	https://thewebminer.com/download	15 de abril de 2019
Quandl	https://www.quandl.com/	15 de abril de 2019
UCI Machine Learning Repository	http://archive.ics.uci.edu/ml/	15 de abril de 2019
CRAWDAD	http://crawdad.org/	15 de abril de 2019
Austin's Open Data Portal	http://data.austintexas.gov	15 de abril de 2019
Chicago Data Portal	http://data.cityofchicago.org	15 de abril de 2019
Data.gov.uk	http://data.gov.uk/	15 de abril de 2019
Data.Medicare.gov	http://data.medicare.gov	15 de abril de 2019
Seattle Open Data Portal	http://data.seattle.gov	15 de abril de 2019
DataSF	http://data.sfgov.org	15 de abril de 2019
Wikipedia: Database download	https://en.wikipedia.org/wiki/Wikipedia:Database_download	15 de abril de 2019
National Institutes of Health	http://ftp.ncbi.nih.gov/	15 de abril de 2019
Stanford Large Network Dataset Collection	http://snap.stanford.edu/data/index.html	15 de abril de 2019
National Archives	https://www.archives.gov/research/alic/tools/online-databases.html	15 de abril de 2019
U.S. Government's open data	http://www.data.gov/	15 de abril de 2019
DBpedia	https://wiki.dbpedia.org/develop/datasets	15 de abril de 2019
Federal Aviation Administration	https://www.faa.gov/data_research/	15 de abril de 2019
Federal Reserve Economic Data	https://fred.stlouisfed.org/	15 de abril de 2019
Build.Kiva	http://build.kiva.org/docs/data/snapshots	15 de abril de 2019
NYC Open Data	https://opendata.cityofnewyork.us/	15 de abril de 2019
City of Vancouver Open Data	https://vancouver.ca/your-government/open-data-catalogue.aspx	15 de abril de 2019
OECD.Stat	https://stats.oecd.org/index.aspx	15 de abril de 2019
Millennium Development Goald Indicators	http://mdgs.un.org/unsd/mdg/Data.aspx	15 de abril de 2019
NOAA Data and Products	https://www.ngdc.noaa.gov/ngdcinfo/onlineaccess.html	15 de abril de 2019
The World Bank Data Catalog	https://data.worldbank.org/	15 de abril de 2019
IMDb Datasets	http://www.imdb.com/interfaces	15 de abril de 2019

Dados.gov	https://dados.gov.pt/pt/datasets/	15 de abril de 2019
Knoema	http://knoema.com	15 de abril de 2019
Berlin Open Data	http://daten.berlin.de/	15 de abril de 2019
Open Data Österreich	https://www.data.gv.at/	15 de abril de 2019
British Columbia DataBC	http://data.gov.bc.ca	15 de abril de 2019
DataShop	https://pslcdatashop.web.cmu.edu/	15 de abril de 2019
ICPSR	https://www.icpsr.umich.edu/icpsrweb/ICPSR/	15 de abril de 2019
data.gov.it	http://www.dati.gov.it	15 de abril de 2019
Open Data Trentino	http://dati.trentino.it	15 de abril de 2019
Network Repository	http://networkrepository.com	15 de abril de 2019
IMF Data	https://www.imf.org/en/Data	15 de abril de 2019
IES NCES	https://nces.ed.gov/	15 de abril de 2019
UK Data Service	https://www.ukdataservice.ac.uk/	15 de abril de 2019
FiveThirtyEight Data	https://data.fivethirtyeight.com/	15 de abril de 2019
BJS All Data Collections	https://www.bjs.gov/index.cfm?ty=dca	15 de abril de 2019
Qlik DataMarket	https://www.qlik.com/us/products/qlik-data-market	15 de abril de 2019
NASA Exoplanet Archieve	https://exoplanetarchive.ipac.caltech.edu/docs/data.html	15 de abril de 2019
UN Comtrade Database	https://comtrade.un.org/	15 de abril de 2019
LFW Face Database	http://vis-www.cs.umass.edu/lfw/	15 de abril de 2019
MS Marco	http://www.msmarco.org/dataset.aspx	15 de abril de 2019
Mldata.org	http://mldata.org/	15 de abril de 2019
Natural History Museum Datasets	https://data.nhm.ac.uk/dataset	15 de abril de 2019
CERN Open Data	http://opendata.cern.ch/	15 de abril de 2019
One Million Audio Cover Images for Research	https://archive.org/details/audio-covers	15 de abril de 2019
London Air	http://www.londonair.org.uk/london/asp/datadownload.asp	15 de abril de 2019
Google Dataset Search	https://toolbox.google.com	15 de abril de 2019
Bureau of Transportation Statistics	https://www.transtats.bts.gov/	15 de abril de 2019
NYC Taxi & Limousine Commission	https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page	15 de abril de 2019

II. Lista de software de visualização informação

Tabela 2. Lista de software para visualização de dados.

Software	Sítio	Último acesso
Excel	https://products.office.com/en/excel	8 de outubro de 2019.
Tableau	https://www.tableau.com/	8 de outubro de 2019.
FusionCharts	https://www.fusioncharts.com/	8 de outubro de 2019.
Knime	https://www.knime.com/	8 de outubro de 2019.
Qlik View	https://www.qlik.com/us/products/qlikview	8 de outubro de 2019.
Highcharts	https://www.highcharts.com/	8 de outubro de 2019.
Datawrapper	https://www.datawrapper.de/	8 de outubro de 2019.
Plotly	https://plot.ly/	8 de outubro de 2019.
Sisense	https://www.sisense.com/	8 de outubro de 2019.
Kibana	https://www.elastic.co/products/kibana	8 de outubro de 2019.
RAW Graphs	https://rawgraphs.io/	8 de outubro de 2019.
D3.js	https://d3js.org/	8 de outubro de 2019.
Power BI	https://powerbi.microsoft.com/en-us/	8 de outubro de 2019.
Data Studio	https://datastudio.google.com/u/0/	8 de outubro de 2019.
Vaex	https://vaex.readthedocs.io/en/latest/	8 de outubro de 2019.
VisIT	https://wci.llnl.gov/simulation/computer-codes/visit	8 de outubro de 2019.
ParaView	https://www.paraview.org/	8 de outubro de 2019.
Mayavi	https://docs.enthought.com/mayavi/mayavi/	8 de outubro de 2019.
HoloViews	http://holoviews.org/	8 de outubro de 2019.
Datashader	https://datashader.org/	8 de outubro de 2019.
Bokeh	https://bokeh.org/	8 de outubro de 2019.
Gnuplot	http://gnuplot.info/	8 de outubro de 2019.
Matplotlib	https://matplotlib.org/	8 de outubro de 2019.
Trelliscope	http://deltarho.org/docs-trelliscope/index.html	8 de outubro de 2019.
Vega-Lite	https://vega.github.io/vega-lite/	8 de outubro de 2019.
Vega	https://vega.github.io/vega/	8 de outubro de 2019.
LatticeExtra	https://cran.r-project.org/web/packages/latticeExtra/index.html	8 de outubro de 2019.
Plot3D	https://cran.r-project.org/web/packages/plot3D/index.html	8 de outubro de 2019.
rgl	https://cran.r-project.org/web/packages/rgl/index.html	8 de outubro de 2019.
Plot3Drgl	https://cran.r-project.org/web/packages/plot3Drgl/index.html	8 de outubro de 2019.
OceanView	https://cran.r-project.org/web/packages/OceanView/index.html	8 de outubro de 2019.
CodeFlower	http://www.redotheweb.com/CodeFlower/	8 de outubro de 2019.
D3plus	http://d3plus.org/	8 de outubro de 2019.
Amazon	https://aws.amazon.com/quicksight/	8 de outubro de 2019.

QuickSight		
DataHero	https://datahero.com/	8 de outubro de 2019.
Data Illustrator	http://www.data-illustrator.com/	8 de outubro de 2019.
dimple	http://dimplejs.org/	8 de outubro de 2019.
JetPack Data	https://www.jetpackdata.com/landing#	8 de outubro de 2019.
amCharts	http://www.amcharts.com/javascript-charts/	8 de outubro de 2019.
p5.js	https://p5js.org/	8 de outubro de 2019.
PlotDB	https://plotdb.com/	8 de outubro de 2019.
Polychart	https://github.com/Polychart	8 de outubro de 2019.
Processing.js	http://processingjs.org/	8 de outubro de 2019.
sigma.js	http://sigmajs.org/	8 de outubro de 2019.
Stata	https://www.stata.com/	8 de outubro de 2019.
Taucharts	https://taucharts.com/	8 de outubro de 2019.
teeChart	https://www.steema.com/	8 de outubro de 2019.
ZoomCharts	https://zoomcharts.com/en/	8 de outubro de 2019.
ZingChart	https://www.zingchart.com/#welcome	8 de outubro de 2019.
Scatterplot.online	https://scatterplot.online/	8 de outubro de 2019.
Chartlr	https://chartlr.com/	8 de outubro de 2019.
Chartbuilder 3.0.5	https://quartz.github.io/Chartbuilder/	8 de outubro de 2019.
Charticulator	https://charticulator.com/	8 de outubro de 2019.
ChartsBin	http://chartsbin.com/	8 de outubro de 2019.
Infocaptor	https://www.infocaptor.com/	8 de outubro de 2019.
Alteryx	https://www.alteryx.com/	8 de outubro de 2019.
Splunk	https://www.splunk.com/	8 de outubro de 2019.
Chartio	https://chartio.com/	8 de outubro de 2019.
Domo	https://www.domo.com/	8 de outubro de 2019.
Sisense	https://www.sisense.com/	8 de outubro de 2019.

III. Lista de HMDs

Tabela 3. Lista de outros HMDs.

HMD	Autónomo	Sítio	Último acesso
HTC Vive	Não.	https://www.vive.com/eu/product/	8 de outubro de 2019.
VIVE Cosmos	Não.	https://www.vive.com/us/product/cosmos/	8 de outubro de 2019.
VIVE Pro	Não.	https://enterprise.vive.com/us/product/vive-pro/	8 de outubro de 2019.
VIVE Pro Eye	Não.	https://enterprise.vive.com/us/product/vive-pro-eye/	8 de outubro de 2019.
VIVE Focus	Sim.	https://enterprise.vive.com/us/product/vive-focus/	8 de outubro de 2019.
VIVE Focus Plus	Sim.	https://enterprise.vive.com/us/product/focus-plus/	8 de outubro de 2019.
Pico G2	Sim.	https://www.pico-interactive.com/eu-en/G2.html	8 de outubro de 2019.
Pico G2 4K	Sim.	https://www.pico-interactive.com/eu-en/G2_4K.html	8 de outubro de 2019.
Valve Index	Não.	https://store.steampowered.com/valveindex	8 de outubro de 2019.
Oculus Quest	Sim.	https://www.oculus.com/quest/	8 de outubro de 2019.
Oculus Rift S	Não.	https://www.oculus.com/rift-s/	8 de outubro de 2019.
Oculus Go	Sim.	https://www.oculus.com/go/	8 de outubro de 2019.
Samsung HMD Odyssey+	Não.	https://www.microsoft.com/en-us/p/samsung-hmd-odyssey/8n2d0nk20p8m	8 de outubro de 2019.
Daydream – Standalone VR	Sim.	https://vr.google.com/daydream/standalonevr/	8 de outubro de 2019.
Fove VR	Não.	https://www.getfove.com/	8 de outubro de 2019.
StarVR One	Não.	https://www.starvr.com/products/	8 de outubro de 2019.
PlayStation VR	Não.	https://www.playstation.com/en-us/explore/playstation-vr/	8 de outubro de 2019.
Pimax VR – 5K Plus Headset	Não.	http://store.pimaxvr.com/shop/product/xpp2-100-0015-5k-plus-headset-25	8 de outubro de 2019.
Pimax VR – 5K XR Headset	Não.	https://store.pimaxvr.com/shop/product/xpp2-100-0005-8k-headset-86	8 de outubro de 2019.
Pimax VR – 8K Headset	Não.	https://store.pimaxvr.com/shop/product/xpp2-100-0009-5k-xr-headset-111	8 de outubro de 2019.
Magic Leap One	Sim	https://www.magicleap.com/magic-leap-one	8 de outubro de 2019.

IV. Lista de outros ADIs

Tabela 4. Lista de outros ADIs, adaptada de Toftedahl, M. (2019). *Which are the most commonly used Game Engines*. Disponível online em: https://www.gamasutra.com/blogs/MarcusToftedahl/20190930/350830/Which_are_the_most_commonly_used_Game_Engines.php (Acedido dia 8 de outubro de 2019).

ADI	Sítio	Último acesso
Gamemaker	https://www.yoyogames.com/gamemaker	8 de outubro de 2019.
Construct	https://www.construct.net/en	8 de outubro de 2019.
Twine	https://twinery.org/	8 de outubro de 2019.
RPG Maker	https://www.rpgmakerweb.com/	8 de outubro de 2019.
Bitsy	https://ledoux.itch.io/bitsy	8 de outubro de 2019.
Pico-8	https://www.lexaloffle.com/pico-8.php	8 de outubro de 2019.
Godot	https://godotengine.org/	8 de outubro de 2019.
Ren'Py	https://www.renpy.org/	8 de outubro de 2019.
CRYENGINE	https://www.cryengine.com/	8 de outubro de 2019.
Gamebryo	http://www.gamebryo.com/	8 de outubro de 2019.
OpenSceneGraph	http://www.openscenegraph.org/	8 de outubro de 2019.

V. Lista de soluções para criar realidade aumentada (RA)

Tabela 5. Lista de soluções para criar RA.

Aplicação	Sítio	Último acesso
A-frame com AR.js	https://aframe.io/examples/ https://github.com/jeromeetienne/ar.js	16 de abril de 2019.
ApertusVR	http://apertusvr.org	16 de abril de 2019.
Appy Pie	https://www.appypie.com/	16 de abril de 2019.
Arize Platform	http://arize.io/index	16 de abril de 2019.
ARCore	https://developers.google.com/ar/	16 de abril de 2019.
ARKit	https://developer.apple.com/arkit/	16 de abril de 2019.
ARToolKit	https://github.com/artoolkit	16 de abril de 2019.
artoolkitX	http://www.artoolkitx.org/	16 de abril de 2019.
Blippar	https://www.blippar.com/	16 de abril de 2019.
DeepAR	https://www.deepar.ai/augmented-reality-sdk	16 de abril de 2019.
DroidAR	https://bitstars.github.io/droidar/	16 de abril de 2019.
EasyAR	https://www.easyar.com/	16 de abril de 2019.
HP Reveal	https://www.hpreveal.com/	16 de abril de 2019.
Kudan AR SDK	https://www.xlsoft.com/en/products/kudan/down_eval.html	16 de abril de 2019.
Layar	https://www.layar.com/	16 de abril de 2019.
MAXST	http://maxst.com/#/	16 de abril de 2019.
UniteAR	https://www.unitear.com/	16 de abril de 2019.
Vuforia	https://www.ptc.com/en/academic-program/academic-products/free-software/vuforia	16 de abril de 2019.
Wikitude	https://www.wikitude.com/	16 de abril de 2019.
XZIMG	https://www.xzimg.com/Home	16 de abril de 2019.
Zappar	https://www.zappar.com/	16 de abril de 2019.
PanicAR	http://panicar.dopanic.com/	16 de abril de 2019.

VI. Protótipo AR (Vuforia)

Para os utilizadores de Android que não querem instalar Zappar e querem à mesma ver o protótipo em 3D, basta instalar a aplicação arDataVisualization, dar permissão à câmara e apontar para a Figura 1. Para quem quiser saber mais detalhes de como fazer algo igual, ver o repositório.

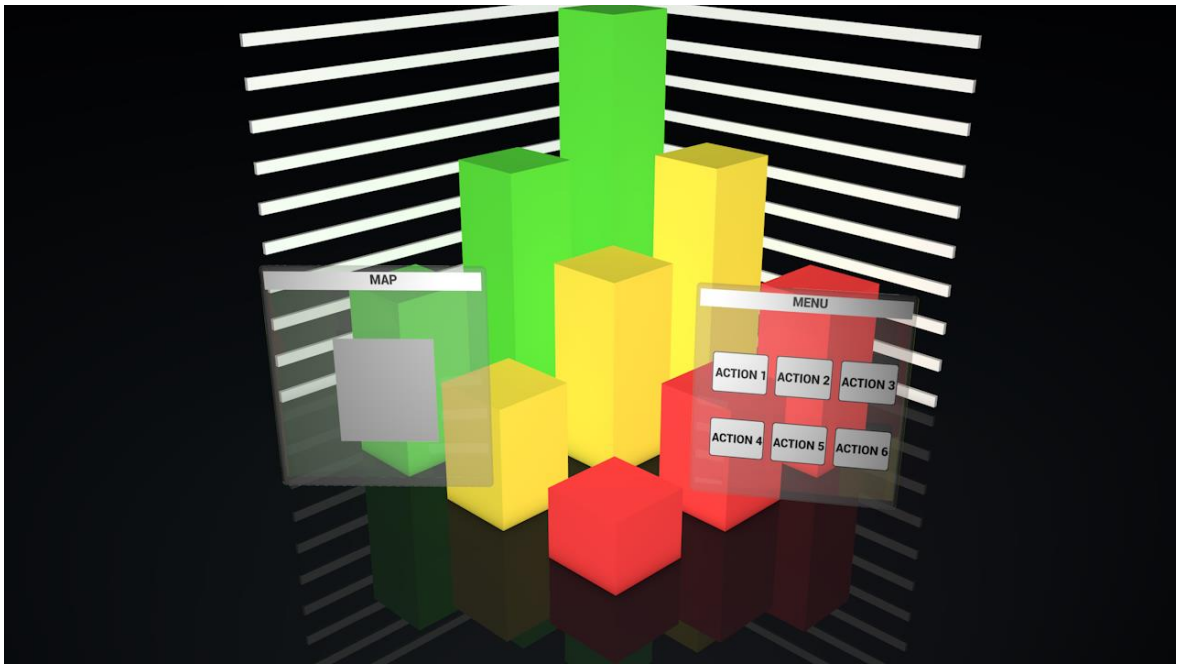

Figura 1. Marcador para onde apontar a câmara e ver o modelo em realidade aumentada.

Repositório: https://github.com/tiago-peres/ar-DataVisualization
Aplicação: https://github.com/tiago-peres/ar-DataVisualization/blob/master/arDataVisualization.apk

VII. *Juntar o conteúdo de vários CSV num único CSV*

O programa utilizado nesta parte foi encontrado na comunidade StackOverflow, mais precisamente no seguinte link: https://stackoverflow.com/a/56295720/5675325 (Acedido dia 16 de outubro de 2019).

```
# -*- coding: utf-8 -*-
"""
Created on Thu Oct  3 21:14:07 2019

@author: tiago
"""

import os
import csv, glob

Dir = r"C:\Users\tiago\Desktop\DataVis1\bar_chart\all_years\data"
Avg_Dir =
r"C:\Users\tiago\Desktop\DataVis1\bar_chart\all_years\data\output"

csv_file_list = glob.glob(os.path.join(Dir, '*.csv'))
print (csv_file_list)

with open(os.path.join(Avg_Dir, 'Output.csv'), 'w', newline='') as f:
    wf = csv.writer(f, lineterminator='\n')

    for files in csv_file_list:
        with open(files, 'r') as r:
            next(r)                  # SKIP HEADERS
            rr = csv.reader(r)
            for row in rr:
                wf.writerow(row)
```

VIII. *Criar amostra a partir do CSV*

Para a criação de amostras no Windows recorremos ao Git Bash e bastou executar versões do seguinte comando

```
shuf -n 10000 BIGFILE.csv > SAMPLEFILE_10000.csv
```

Neste caso estamos a criar uma amostra do ficheiro BIFFILE.csv com dez mil linhas e guardar num ficheiro, na mesma localização, com nome SAMPLEFILE_10000.csv.

IX. *Adicionar e remover linhas de um CSV*

Para adicionar cabeçalhos a um CSV no Windows recorremos ao Git Bash e bastou executar versões do seguinte comando

```
sed 1i"year,month,temperature" output.csv > new_output.csv
```

Neste caso estamos a adicionar cabeçalhos *year*, *month*, *temperature* ao ficheiro output.csv guardar num ficheiro, na mesma localização, com nome new_output.csv.

Por outro lado, para remover os cabeçalhos, executamos o seguinte comando (a diferença para o comando anterior é a não necessidade de especificar o conteúdo da linha e o *i*, de *insert*, deu lugar a *d*, de *delete*)

```
sed 1d output.csv > new_output.csv
```

X. Ordenar um CSV por coluna

Para ordenar um CSV no Windows recorremos ao PowerShell e bastou executar versões do seguinte comando

```
Import-Csv sample_1000.csv | sort year | Export-Csv
sorted_sample_1000.csv -NoTypeInformation
```

Neste caso estamos a importar o CSV sample_1000.csv, a ordenar por *year*, e a guardar o resultado num ficheiro, na mesma localização, com nome sorted_sample_1000.csv. Acontece que o ficheiro resultante deste comando aparece com aspas e com uma linha a mais além do cabeçalho

```
year,"month","temperature"
2009,"07","20.500000000000004"
2009,"06","18.944444444444443"
2009,"09","-12.055555555555555"
2009,"03","-4.333333333333333"
```

Para remover as aspas recorremos ao seguinte comando

```
set-content 1.csv ((get-content 1.csv) -replace '"')
```

Para remover linhas no CSV explicamos na secção X, sendo que o resultado fica

```
1936,08,15.111111111111111
1940,04,1.3333333333333326
1943,02,18.722222222222221
1944,06,29.833333333333332
1944,08,14.166666666666666
1945,01,15.555555555555555
1946,08,20.722222222222221
1948,03,15.277777777777779
```

XI. CSVReader para o Unity

O CSVReader utilizado foi criado por Teemu Ikonen. Basicamente, são examinadas as expressões regulares, lê o ficheiro CSV colocado na pasta "Assets/Resources" e converte como um dicionário para uso posterior. (Disponível online em: (https://bravenewmethod.com/2014/09/13/lightweight-csv-reader-for-unity/ (Acedido dia 6 de setembro de 2019)).

```
using UnityEngine;
using System;
using System.Collections;
using System.Collections.Generic;
using System.Text.RegularExpressions;

public class CSVReader
{
    static string SPLIT_RE = @",(?=(?:[^""]*""[^""]*"")*(?![^""]*""))";
    static string LINE_SPLIT_RE = @"\r\n|\n\r|\n|\r";
    static char[] TRIM_CHARS = { '\"' };

    public static List<Dictionary<string, object>> Read(string file)
    {
        var list = new List<Dictionary<string, object>>();
        TextAsset data = Resources.Load(file) as TextAsset;

        var lines = Regex.Split(data.text, LINE_SPLIT_RE);

        if (lines.Length <= 1) return list;

        var header = Regex.Split(lines[0], SPLIT_RE);
        for (var i = 1; i < lines.Length; i++)
        {

            var values = Regex.Split(lines[i], SPLIT_RE);
            if (values.Length == 0 || values[0] == "") continue;

            var entry = new Dictionary<string, object>();
            for (var j = 0; j < header.Length && j < values.Length; j++)
            {
                string value = values[j];
                value =
value.TrimStart(TRIM_CHARS).TrimEnd(TRIM_CHARS).Replace("\\", "");
                object finalvalue = value;
                int n;
                float f;
                if (int.TryParse(value, out n))
                {
                    finalvalue = n;
                }
                else if (float.TryParse(value, out f))
                {
                    finalvalue = f;
                }
                entry[header[j]] = finalvalue;
            }
            list.Add(entry);
        }
        return list;
    }
}
```

Acontece que este CSVReader usa List<Dictionary<string, object>>, ou seja, se não existe cabeçalho no CSV, as chaves do dicionário serão muito provavelmente nulas. Assim, com certos ajustes criámos uma versão alternativa do método que retorna uma lista simples de objetos <List<List<object>>, removendo o código que preenche os cabeçalhos.

```
using UnityEngine;
using System;
using System.Collections;
using System.Collections.Generic;
using System.Text.RegularExpressions;

public class CSVReaderNoHeader : MonoBehaviour
{
    static string SPLIT_RE = @",(?=(?:[^""]*""[^""]*"")*(?![^""]*""))";
    static string LINE_SPLIT_RE = @"\r\n|\n\r|\n|\r";
    static char[] TRIM_CHARS = { '\"' };

    public static List<List<object>> Read(string file)
    {
        var list = new List<List<object>>();
        TextAsset data = Resources.Load(file) as TextAsset;
        var lines = Regex.Split(data.text, LINE_SPLIT_RE);

        if (lines.Length <= 1) return list;

        for (var i = 0; i < lines.Length; i++)
        {

            var values = Regex.Split(lines[i], SPLIT_RE);
            if (values.Length == 0 || values[0] == "") continue;
            var entry = new List<object>();

            for (var j = 0; j < values.Length; j++)
            {
                string value = values[j];
                value =
value.TrimStart(TRIM_CHARS).TrimEnd(TRIM_CHARS).Replace("\\", "");
                object finalvalue = value;
                int n;
                float f;
                if (int.TryParse(value, out n))
                {
                    finalvalue = n;
                }
                else if (float.TryParse(value, out f))
                {
                    finalvalue = f;
                }
                entry.Add(finalvalue);
            }
            list.Add(entry);
        }

        return list;
    }
```

```
}
```

Supondo que o CSV tem nome 1.csv e tem o seguinte formato:

```
Examplo 1.csv na pasta Resources.
1   2009,12,35
2   2010,11,10
3   2014,10,7,5
4   2018,10,2,333
```

Podemos ler da seguinte forma:

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using System;

public class BarSample1 : MonoBehaviour
{

    void Awake()
    {
        List<List<object>> data = CSVReaderNoHeader.Read("1");

        for (var i = 0; i < data.Count; i++)
        {
            double month = (double)(int)data[i][1];
            string month_string = month.ToString();

            print($"month:{month} ///////END");
            print($"month_string:{month_string} ///////END");

        }


    }
}
```

XII. *Data-Room*

Para a construção deste ambiente, de salientar a relevância da especialização *Virtual Reality (VR) App Development* da Universidade de San Diego, Califórnia (https://www.edx.org/professional-certificate/virtual-reality-vr-app-development (Acedido dia 1 de março de 2019)). O código criado nesta parte pode ser encontrado em Anexo, secção XIII.

1. **Configurar o Unity para Google Cardboard.**

Assim que o novo projeto Unity carrega, o *default* é um projeto para computadores e nós queremos uma aplicação. Para mudar isto vamos a File > Build Settings (Figura 2).

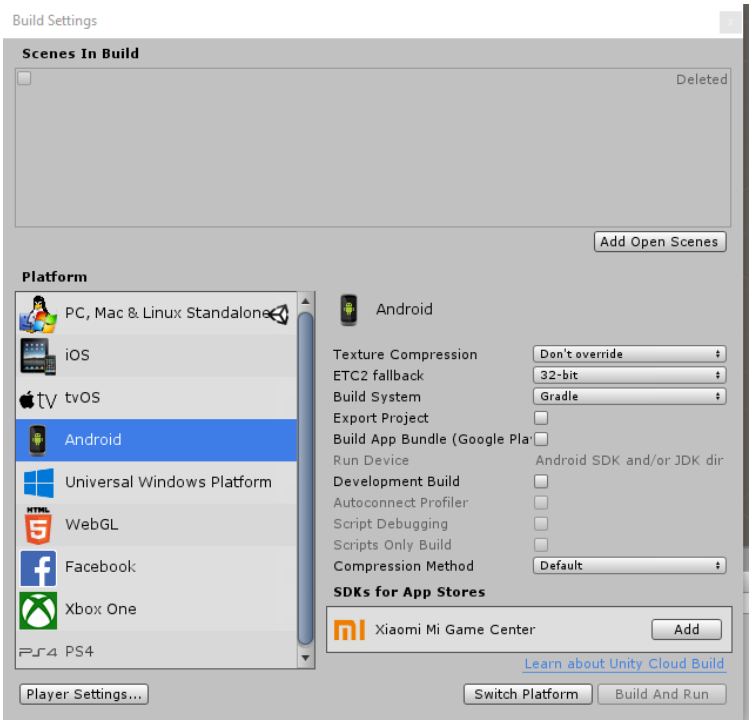

Figura 2. *Build Settings* no Unity.

Apenas precisamos de clicar em "Switch Platform" (Figura 2) para garantir que estamos a construir para Android.

2. **Configurar suporte de realidade virtual que permite "olhar" para diferentes direções e receber *input* do utilizador (botão do Google Cardboard);**

Neste caso poderíamos interagir com objetos para os quais estamos a olhar ou para onde a cabeça está direcionada. Por ser mais simples de implementar, optámos pela segunda opção. Assim, quando o utilizador movimenta a cabeça, significa que o telemóvel no HMD também se move e há sensores que podem detectar a direção do telemóvel para que possamos atualizar a imagem corretamente.

Para além disso, para que o possamos interagir com o ambiente intuitivamente, precisamos de feedback. Pensando sobre uma página web, quando passamos com o ícone do rato por cima de um botão, por vezes o mesmo muda de cor (assumindo que o botão tira partido do selecionador CSS :hover). Também temos a opção de despoletar uma ação após

clicar no botão, por exemplo, mas o facto do botão mudar de cor demonstra ao utilizador que existe uma ação possível clicando no botão. Esse feedback é relevante para o utilizador e é o que procuramos implementar neste ponto. De notar que o HMD que estamos a utilizar tem um botão que permite pressionar e despoletar uma ação (funcionará de forma idêntica ao clicar com um rato) e, dessa forma, não precisamos de recorrer, por exemplo, ao método "olhar e esperar" (apontar o retículo para um objeto com o qual podemos interagir e aguardar dois ou três segundos).

Para implementar estas funcionalidades precisamos do prefab GVR Editor Emulator https://developers.google.com/vr/reference/unity/prefab/GvrEditorEmulator (acedido dia 3 de março de 2019). Basta ir ao seguinte link, fazer download do Google VR SDK for Unity https://developers.google.com/vr/develop/unity/download (acedido dia 3 de março de 2019) e arrastar para dentro do Unity e importar (clicar no botão "Import" possível de ver na Figura 3).

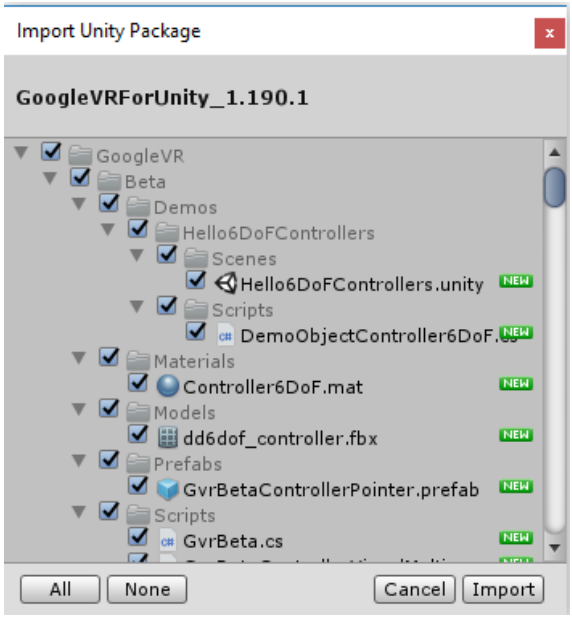

Figura 3. Importar Google VR para Unity.

Quando finalizar a importação, podemos ver em Assets uma pasta de nome Google VR (Figura 4).

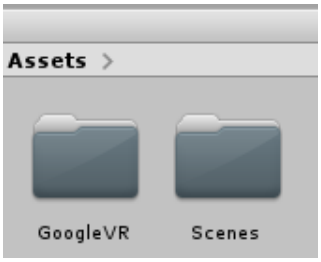

Figura 4. GoogleVR importado.

Como queremos construir para Cardboard, precisamos de ir a Edit > Project Settings > Player > XR Settings e selecionar "Virtual Reality Supported". De notar que aparece uma mensagem de aviso "Must add at least one Virtual Reality SDK". Como já

adicionámos o Google VR aos Assets, ao clicar no ícone + e escolhemos Cardboard (Figura 5).

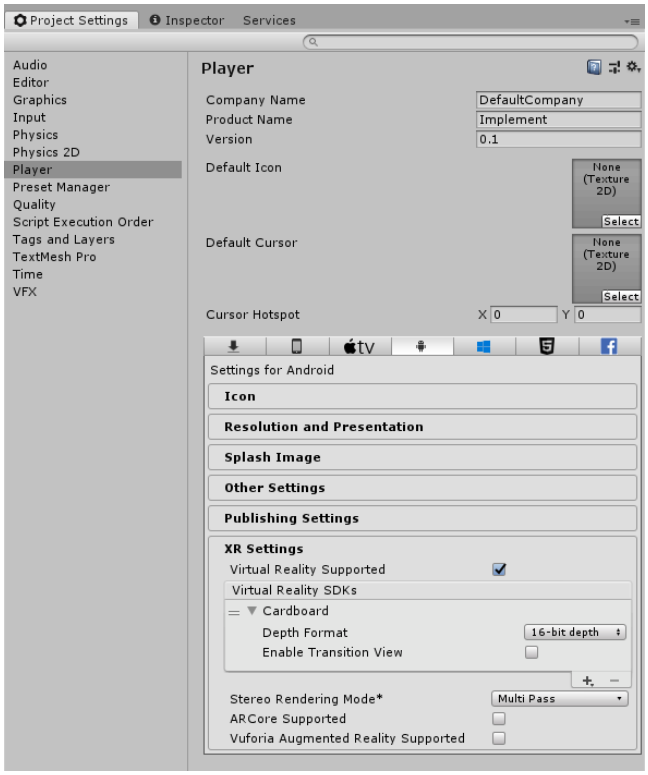

Figura 5. *Player Settings* no Unity.

Podemos encontrar o prefab em Assets > Google VR > Prefabs > GvrEditorEmulator e copiar e colar para a nossa cena (Figura 6).

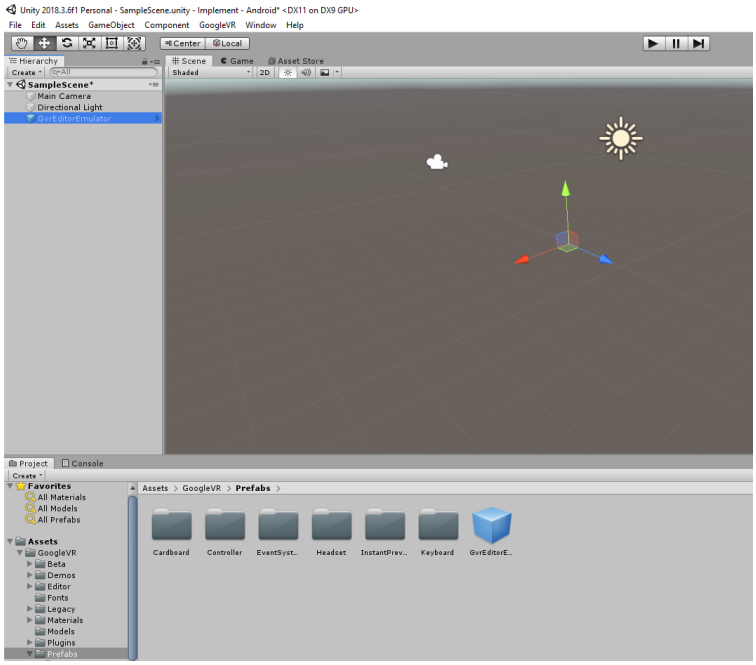

Figura 6. GvrEditorEmulator.

Como o Virtual Reality SDK Cardboard não é suportado no Editor Play Mode, precisamos de fazer um "Build and run" num dispositivo suportado. Para testar o que vai sendo construído no telemóvel Android seguimos o tutorial presente em https://unity3d.com/learn/tutorials/topics/mobile-touch/building-your-unity-game-android-device-testing (acedido dia 3 de março de 2019) e o resultado é possível de ver na Figura 7.

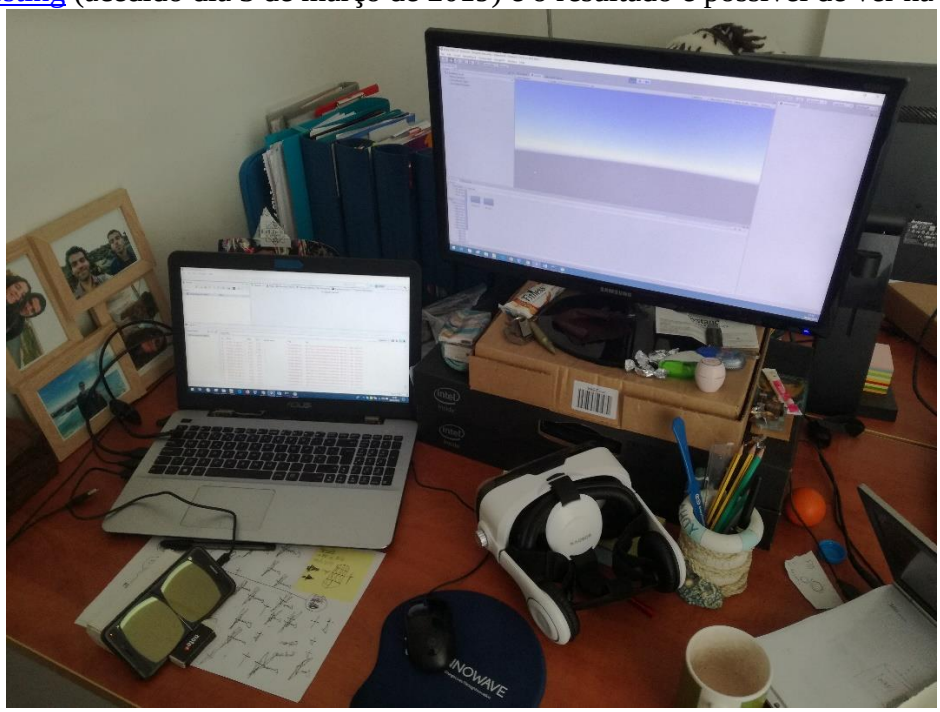

Figura 7. Ambiente de desenvolvimento.

Assim à medida que desenvolvemos as funcionalidades necessárias podemos testar diretamente no telemóvel e ver informações no console do Unity, assim como outras informações que estão a acontecer no sistema operativo do telemóvel através do Android Device Manager. De notar que como podemos ver na Figura 8, temos o suporte de realidade virtual configurado.

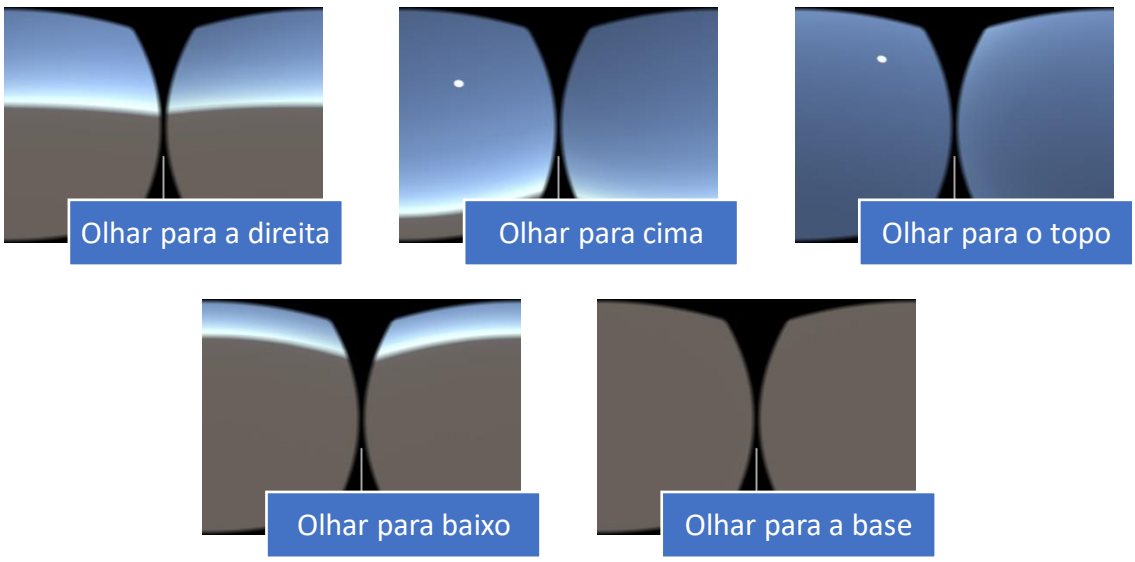

Figura 8. Suporte de realidade virtual configurado.

3. **Configurar um *gaze system*;**

Por *gaze system* definimos a interação com objetos para onde a cabeça está direcionada (tirando partido da funcionalidade até então implementada). No caso deste *gaze system* que queremos implementar vamos usar um retículo no centro do ecrã para sabermos para onde estamos a "olhar" (semelhante ao ícone do rato no computador), se podemos interagir com o objeto em questão (para tal vamos recorrer a mudança de cor no retículo) e se o objeto é o mesmo ou não.

Para interagir com um objeto, tirámos partido do já implementado no ponto 2 e de colisões. Queremos apartir do ponto inicial da câmara, lançar um raio que vai infinitamente para a frente e saber se o mesmo está a colidir com um objeto. Para processar este sistema, criámos o script *GazeSystem.* No mesmo, temos a função *ProcessGaze*() onde recorremos

- ao objeto *Ray* (https://docs.unity3d.com/ScriptReference/Ray-ctor.html (acedido dia 16 de abril de 2019)) para receber a posição inicial da câmara e a posição para a frente;
- a uma variável do tipo *RaycastHit* (https://docs.unity3d.com/ScriptReference/RaycastHit.html (acedido dia 16 de abril de 2019)) para armazenar a informação do raio;
- a uma função *Physics.Raycast* (https://docs.unity3d.com/ScriptReference/Physics.Raycast.html (acedido dia 16 de abril de 2019)) para criar o raio. Este raio está a considerar se é possível interagir com um objeto, se o objeto é o mesmo e mudar a cor do retículo em conformidade (para mudar a cor do retículo recorremos ao método *SetReticleColor*()) ;
- ao método de debug para raios, *Debug.DrawRay*, (https://docs.unity3d.com/ScriptReference/Debug.DrawRay.html (acedido dia 16 de abril de 2019)) para podermos ver para onde o raio está a apontar.

A função *ProcessGaze*() está a ser chamada dentro de *Update*() simplesmente porque queremos estar sempre a saber para onde estamos a "olhar".

Da mesma forma que o ícone do rato pode assumir qualquer forma, o retículo também. Nós optámos por utilizar uma esfera pois o mesmo não vinha especificado nos requerimentos uma forma (Figura 9). Para perceber quando o utilizador clica no botão recorremos ao método *CheckForInput*(). Neste último precisamos de considerar quando o botão foi pressionado, se o mesmo continua a ser pressionado e se o utilizador largou o botão.

Figura 9. Retículo.

Para que possamos saber quando começámos, quando estamos e quando parámos de "olhar para um objeto" e como não queremos interagir com todos os objetos, criámos um script com a classe *GazeableObject* e vários métodos (incluindo três que consideram quando o botão do HMD é pressionado, continua a ser pressionado e foi largado). Assim, como os objetos respondem ou podem responder à interação, podemos criar *currentGazeObject* do tipo *GazeableObject* para ter acesso a todas as funções do *GazeableObject* (Figura 10).

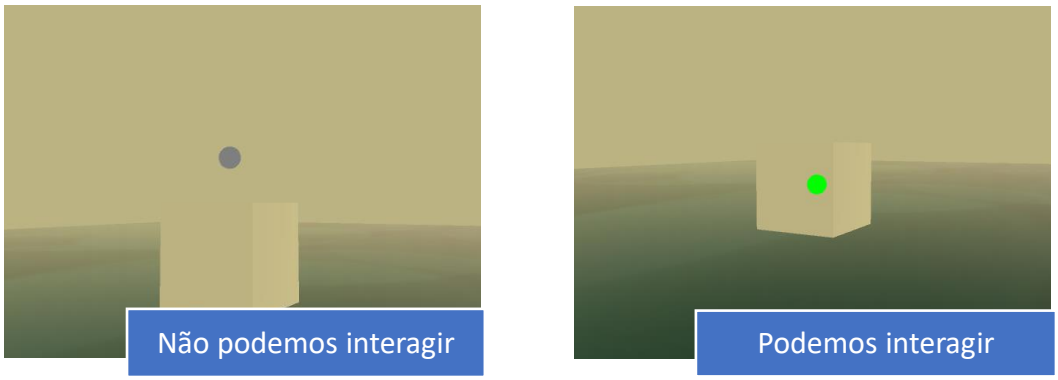

Figura 10. Interação com o retículo.

4. **Criar uma interface de utilizador *gazeable* (menu);**

No ambiente em questão temos 360 graus em redor, ou seja, não há um espaço dedicado onde podemos colocar o menu tal como aquando da construção de uma página web onde o menu está forçado a um ecrã de duas dimensões. Para começar criámos um painel UI com botões. O sistema padrão do Unity tem muito código para lidar com esses tipos de *input*, como, por exemplo, cliques do rato, toques no ecrã (para dispositivos móveis), mas não funciona com ambientes imersivos virtuais de forma intuitiva. Assim, optámos por substituí-lo pelo nosso próprio sistema.

Para que os botões funcionem bem com o *gaze system*, temos de adicionar a cada um componente *Box Collider* (Figura 11). De notar que as letras no menu aparecem desfocadas; podemos alterar isso no objeto *Canvas* que tem um componente *Canvas Scaler* e ao alterar a variável *Dynamic Pixels Per Unit* de 1 para 10 obtemos o resultado da Figura 12.

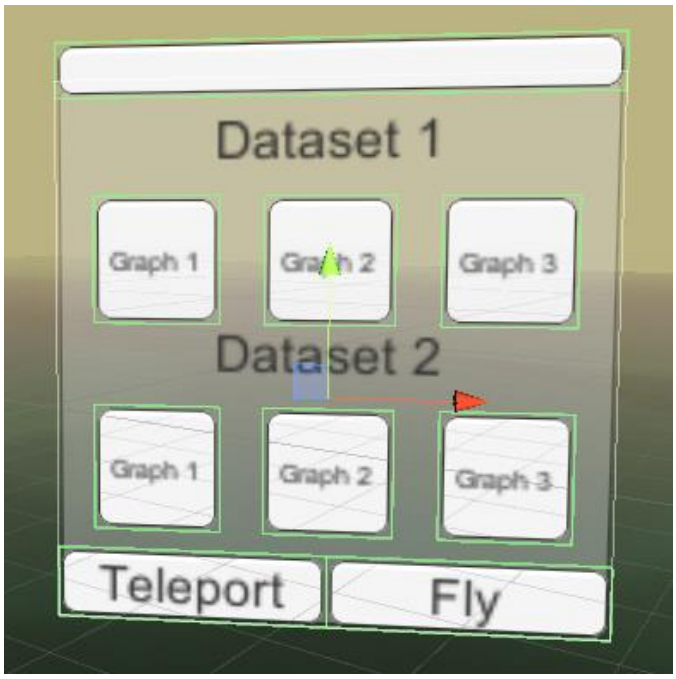

Figura 11. Menu interativo.

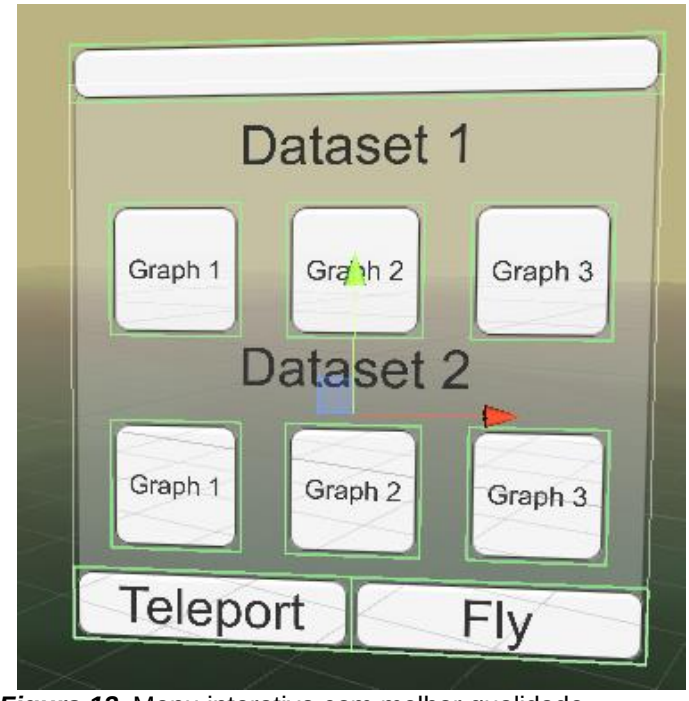

Figura 12. Menu interativo com melhor qualidade.

Para que os botões possam mudar de cor quando carregamos (*OnPress*) num deles (e que desativem caso carreguemos num outro) criámos o script *GazeableButton*. A classe *GazeableButton*, que está a estender *GazeableObject*, tem um método *SetButtonColor* onde podemos passar qualquer cor que quisermos. Como este método recorre a *GetComponent<Image>().color*, estaríamos perante uma exceção para o caso de não existir Image (null.color). Como este script só faz sentido num botão com uma imagem, impusemos uma imagem a este objeto com *[RequireComponent(typeof(Image))]*. Assim, sempre que o

Unity não conseguir adicionar automaticamente o componente *Image*, também não deixará adicionar este componente (Figura 13).

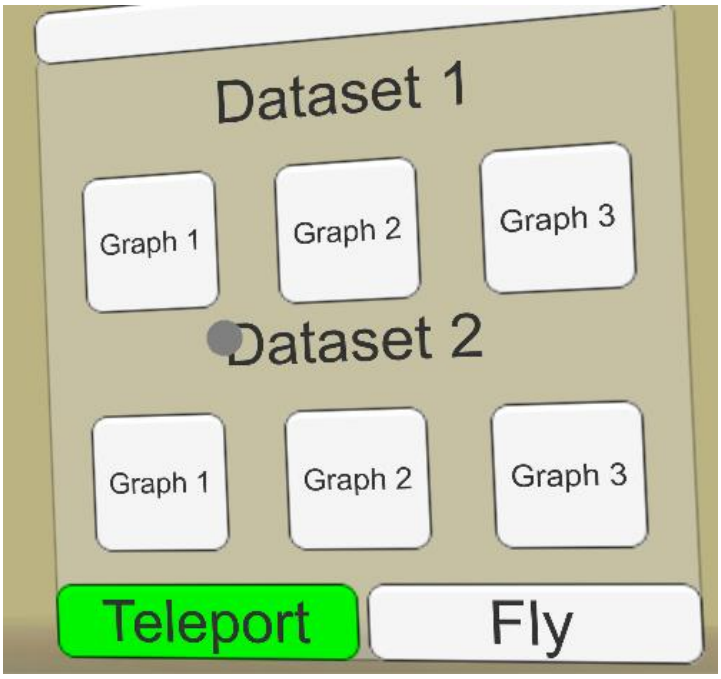

Figura 13. Menu interativo em acção.

Neste momento, ainda que o botão mude de cor, se clicarmos num outro, ambos ficam com a cor verde. Como solução, criámos um script de nome *VRCanvas*; cada botão utiliza uma referência a *VRCanvas* e irá informar de quando foi clicado no método *SetActiveButton()*. Como vemos na Figura 14, o botão é um filho da Canvas; assim, no método *Start()* da classe *GazeableButton* usamos *GetComponentInParent<VRCanvas>();* e eventualmente vai encontrar a *VRCanvas*. De salientar ainda o método *LookAtPlayer()* que garante faz com que o menu fique sempre virado para o visualizador de informação independentemente da posição (algo útil aquando da implementação das funcionalidade de locomoção e permissão para arrastar o menu).

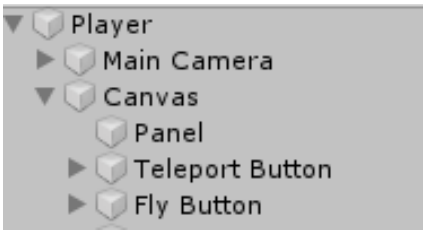

Figura 14. Hierarquia no Unity.

5. **Adicionar locomoção;**

Mover-se nestes ambientes raramente é comparável ao modo como as pessoas se movimentam na vida real, embora existam alguns dispositivos que tentam simular movimentos reais. Quando pensamos em locomoção, temos necessariamente de considerar potenciais causas de enjoo. O enjoo acontece porque os olhos do utilizador acreditam que o utilizador está a mover-se, mas o ouvido interno não detecta movimento [60].

Como vemos nos botões na Figura 13, implementámos dois tipos de locomoção: "Teletransport" e "Fly". Teletransporte permite aos utilizadores ir instantaneamente de um local para outro no chão à distância de um clique, evitando a aceleração suave ou movimento contínuo que geralmente causa enjoo. Por sua vez o voar permite que o utilizador se movimente no espaço de forma mais realista, imersiva, livre (podemos sair do chão) e possa, posteriormente, visualizar os dados de diversos ângulos.

A forma que desenhámos foi através de um script *Player* que controla o estado ativo em que visualizador de informação se encontra. Como muitas classes terão de aceder a este estado ativo do utilizador, utilizámos padrão de design *singleton*, que envolve uma referência estática a este objeto. Ao criar uma instância pública e estática *Player*, podemos aceder à mesma em qualquer parte do código. De notar ainda que na classe *Player* recorremos ao método *Awake* (https://docs.unity3d.com/ScriptReference/MonoBehaviour.Awake.html (acedido dia 18 de abril de 2019)) para iniciar a instância e garantir que existe apenas um *Player*.

Através do enum *InputMode* definimos, de forma conveniente, os modos possíveis atuais para o *Player* (NONE, TELEPORT, FLY) a serem programados. Com isto podemos alterar o estado ativo do *Player* manualmente (Figura 15). Para isso vamos criar a uma nova classe (*ModeButton*) que extende *GazeableButton* e vai mudar o modo ativo do utilizador.

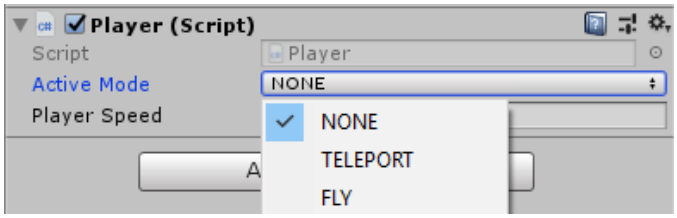

Figura 15. Modo ativo de Player.

Este componente *ModeButton* é adicionado a cada botão que controle o modo ativo do *Player*, componente este que requer especificação de que modo o botão trata; para o caso de voar, selecionamos no *dropdown* "FLY". Assim podemos ver o estado ativo do *Player* a alterar em conformidade aquando o botão é clicado (Figura 16).

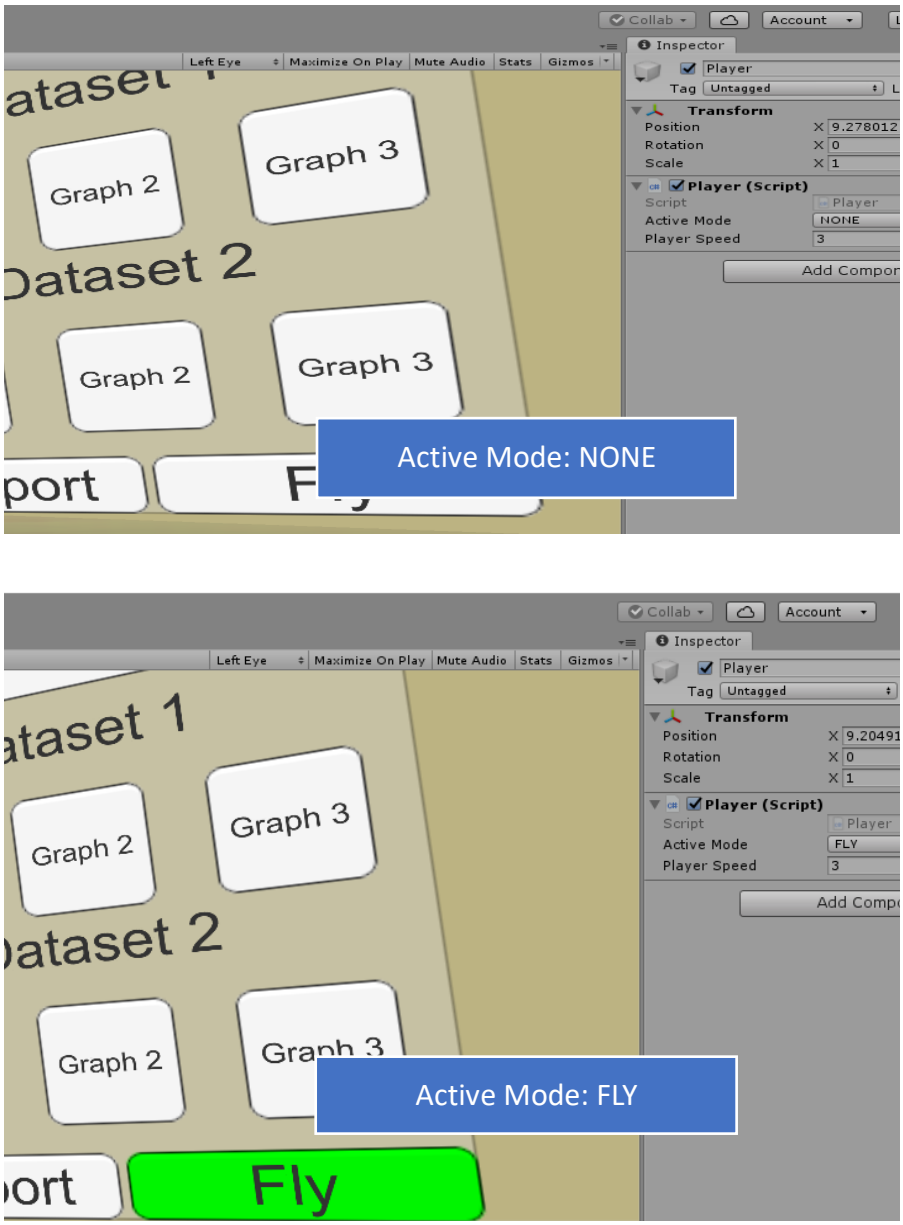

Figura 16. Modo ativo definido para FLY.

Com esta infraestrutura criada, podemos implementar o teletransporte e o voar. Para teletransporte, criámos um script *Floor*, que estende *GazeableObject*, e vai considerar quando o raio está direcionado para o objeto que contém este script como componente (que será o chão, ao qual nomeámos de *Plane*). Podemos ver nas Figuras 17, 18 e 19 que o *Plane* tem o componente *Floor* e, consequentemente, ao "olharmos" para o chão o retículo fica verde e após clicar no botão *Teleport* o modo ativo do *Player* fica "TELEPORT" e ao clicar no *Plane* a posição do *Player* altera tipo teletransporte.

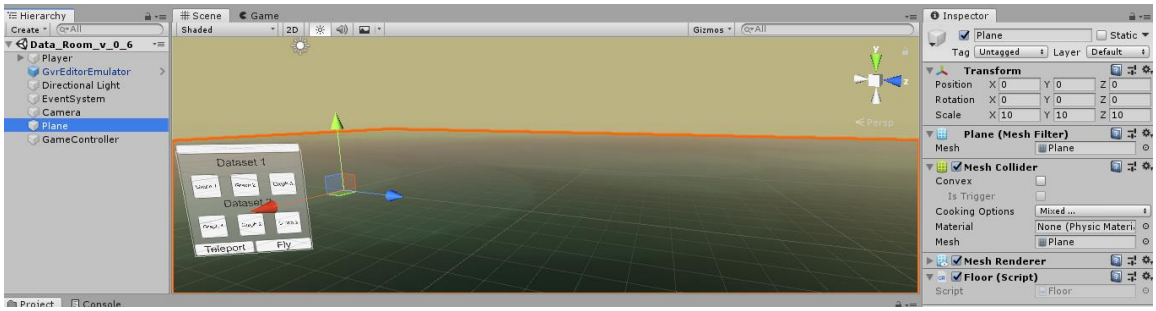

Figura 17. Chão permite teletransporte.

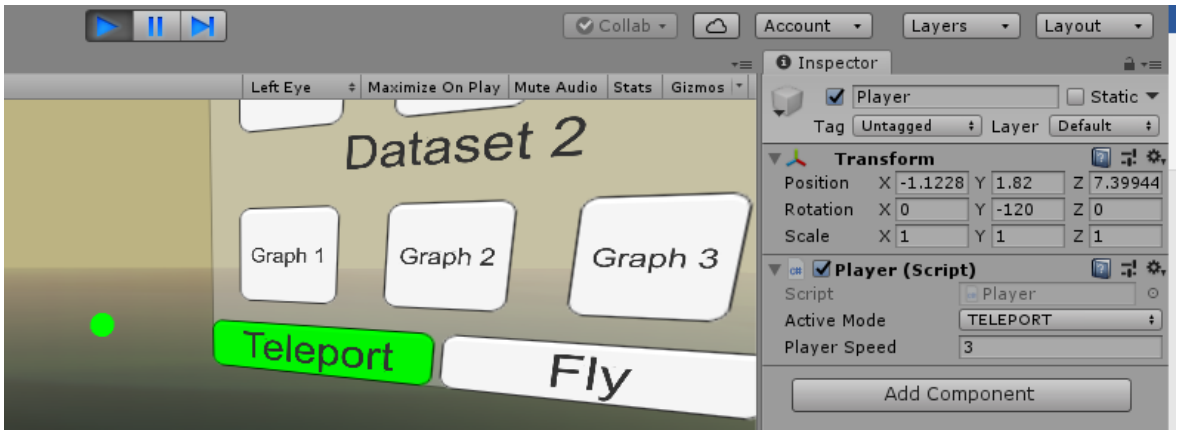

Figura 18. Posição inicial com teletransporte.

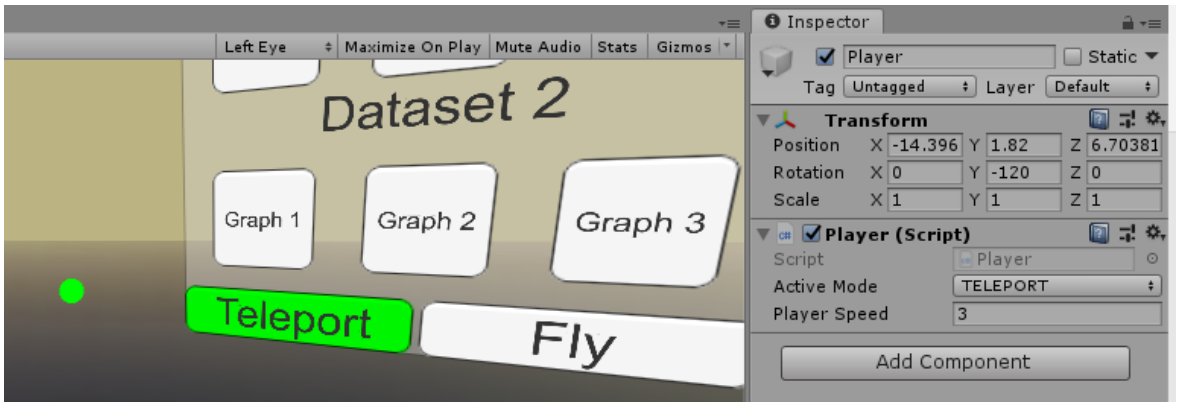

Figura 19. Variação da posição com teletransporte.

Por sua vez, a funcionalidade de voar (deslocação somente para a frente) foi implementada através da modificação de *Player* adicionando o método *TryWalk()*. Dentro deste método temos *Camera.main* que funciona de forma semelhante a *Player.instance*. *Time.deltaTime* é utilizado para que o movimento seja independente do número de *frames*, ou seja, obtemos uma experiência mais homogénea quer em *smartphones* mais rápidos quer em *smarphones* mais lentos.

6. **Implementar um mini-mapa (técnica de orientação);**

Um mini-mapa é um bom sistema de orientação para quem visualiza informação num espaço onde se pode deslocar porque demonstra uma visão *top-down* do espaço. Para isso basta criar uma câmara em que fazemos variar o y a partir de um centro que nos pareça

óptimo, rodar 90º em x, definir "Target Eye" como "None (Main Display)", criar uma esfera pequena que transmita a informação onde o utilizador se encontra e criar um painel de utilizador onde transmitimos a informação da câmara através de *RenderTexture*, Figura 20 (https://docs.unity3d.com/ScriptReference/RenderTexture.html (acedido dia 11 de junho de 2019)), colocando esta textura na propriedade "Target Texture" do componente da câmara. Para que a textura possa ser aplicada ao painel, criámos um material (Figura 21) e o resultado é possível de ver na Figura 22.

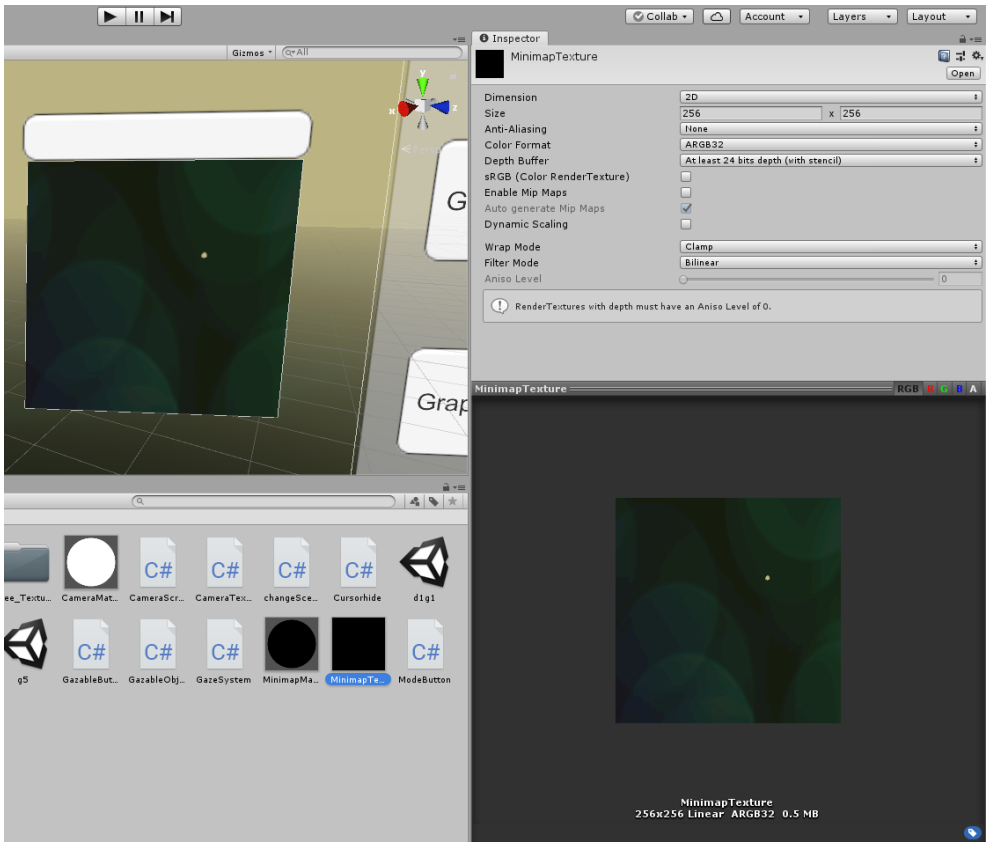

Figura 20. *RenderTeture*.

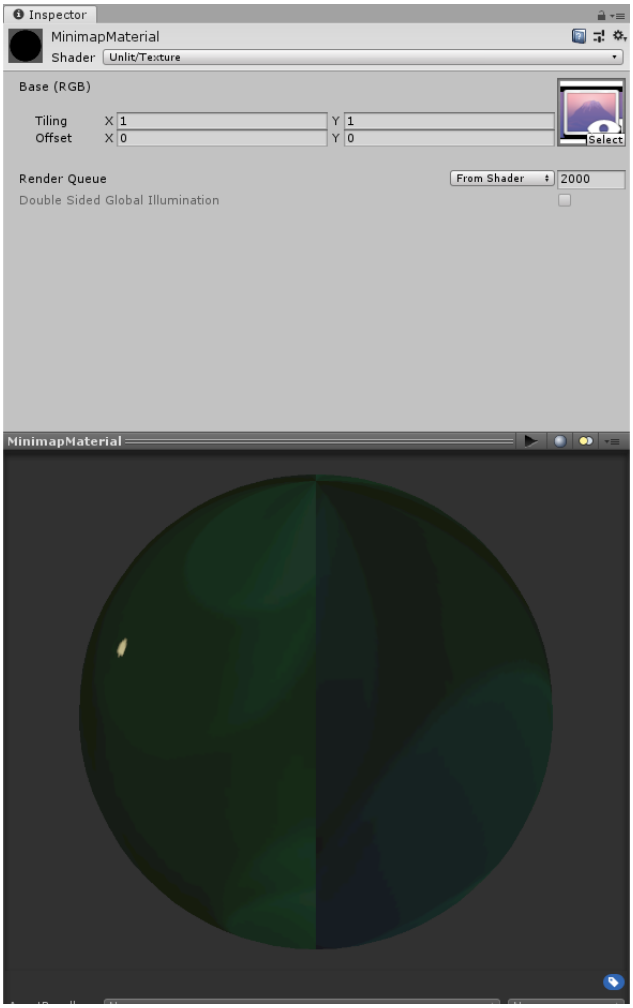

Figura 21. Material.

Figura 22. Efeito resultante da aplicação do material, em acção.

7. **Tornar os painéis de interface do utilizador (menu e mini-mapa) arrastáveis;**

Para que os painéis de interface do utilizador não ocluam ou obstruam outros objetos, permitir que os utilizadores arrastem os mesmos adiciona um nível de personalização pois o mesmo pode desejar que o painel fique num lugar diferente quando estiver num novo local. Com isto em mente e considerando que os painéis tem um componente branco em cima (Figura 23), adicionámos ao componente o script *DragZone* para poder arrastar o painel para onde o utilizador está a "olhar".

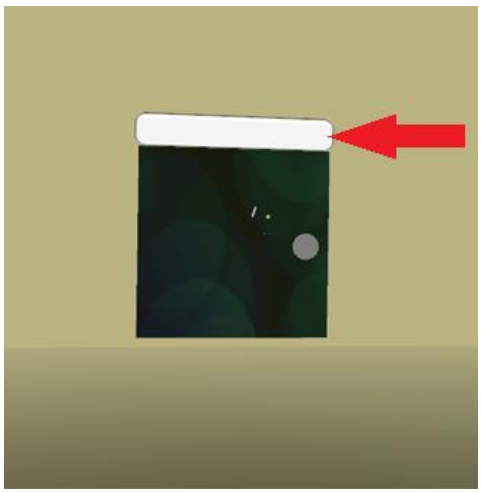

Figura 23. Painéis arrastáveis.

8. **Importar a biblioteca Graph and Chart e criar uma cena com GameObjects 3D (gráficos da biblioteca Graph and Chart);**

Após comprar este item na Unity Asset Store, para importar bastou simplesmente ir a Window > Asset Store > Import > Import (Figura 24).

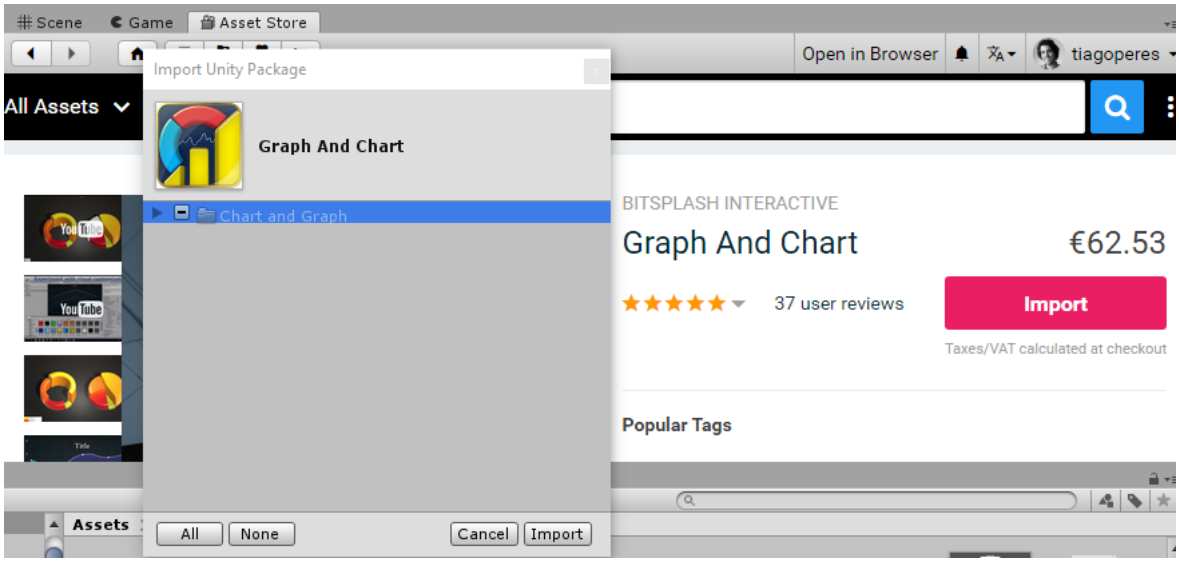

Figura 24. Importar asset *Graph And Chart*.

Para adicionar um gráfico 3D fomos a Assets > Chart and Graph > Themes > 3d > Bar, escolher um dos temas, abrir o Preset, copiar o objeto Bar3DMultiple e colar na cena (Figura 25 e Figura 26). Uma outra forma de fazê-lo é através do menu Tools que aparece após importação do asset.

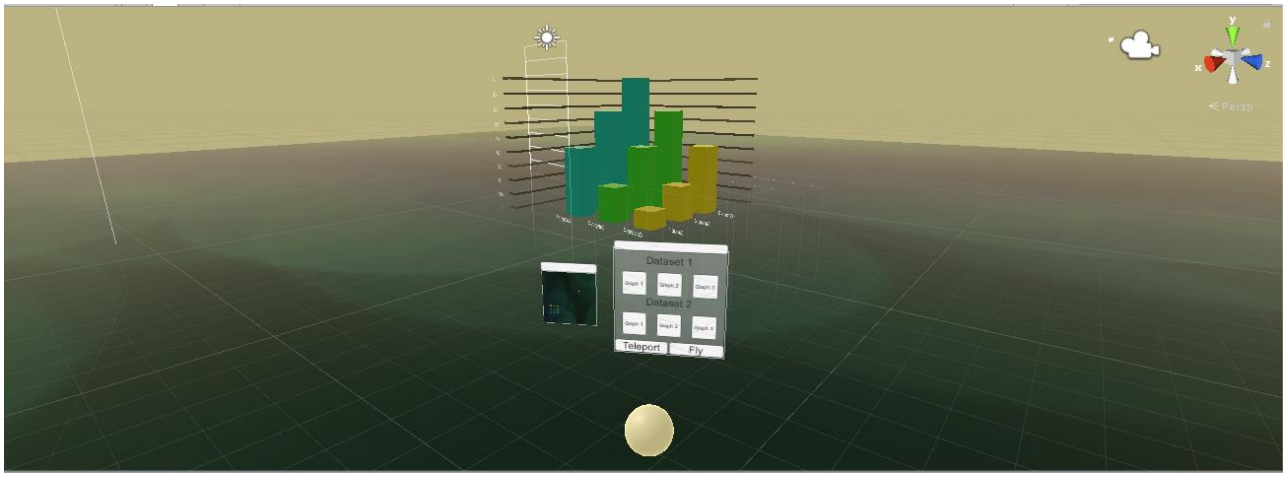

Figura 25. Gráfico de barras 3D em cena.

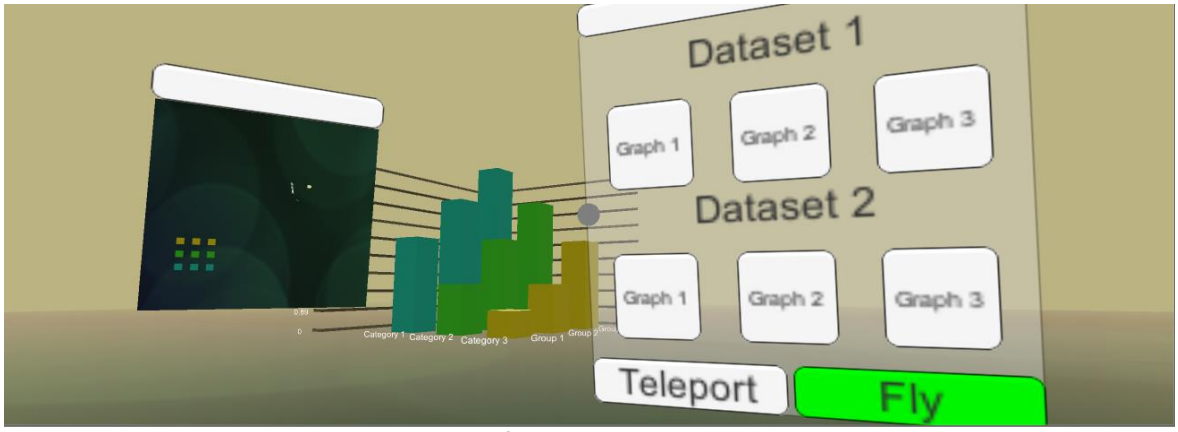

Figura 26. Gráfico de barras 3D em acção.

Ao fazer a *build* do .apk e abrir num Huawey PRA-LX1, com versão Android 8.0.0 e giroscópio, obtemos o que está na Figura 27.

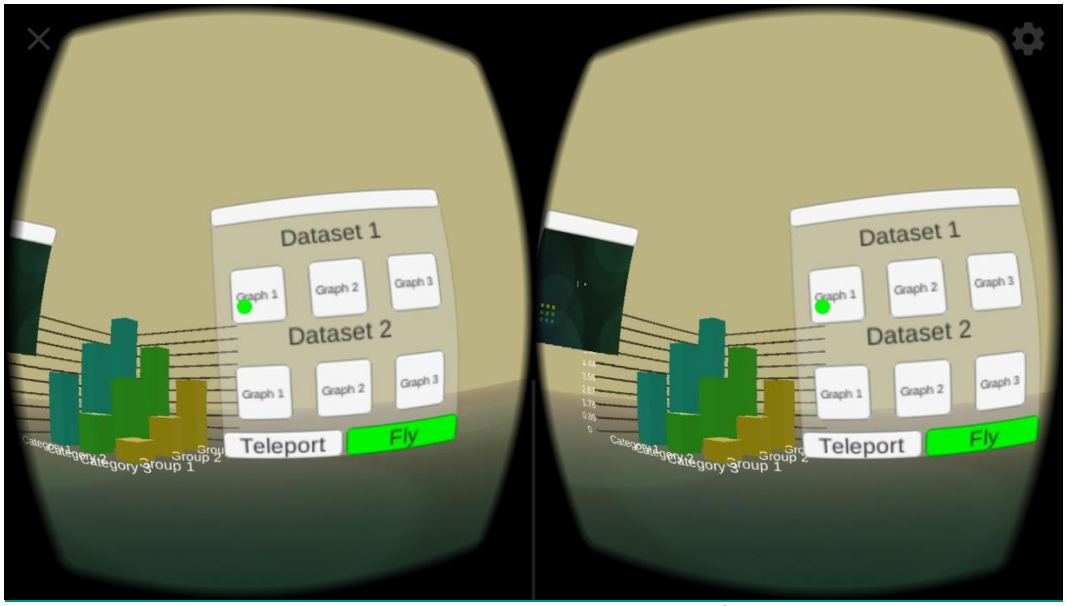

Figura 27. Ambiente visto no telemóvel.

9. **Configurar funcionalidade no interface de utilizador;**

Assim que o utilizador abre a aplicação é redirecionado para uma cena onde existem apenas o mini-mapa e o menu com os quais o mesmo pode interagir (Figura 28). No entanto, sempre que o utilizador clica num dos seis botões centrais do menu, através de uma adaptação do script de *Player* introduzindo funções como o *GoTo1* que muda a cena para a primeira codificação visual, exibimos uma nova cena semelhante (contendo o mini-mapa e menu) mas com o gráfico e com a cor do botão clicado alterada para verde (Figura 29).

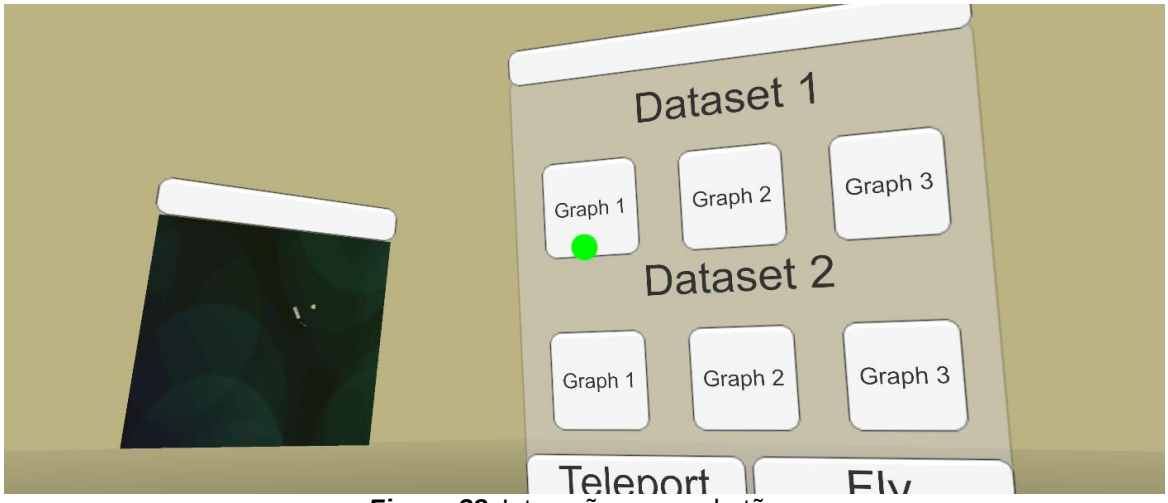

Figura 28. Interação com os botões.

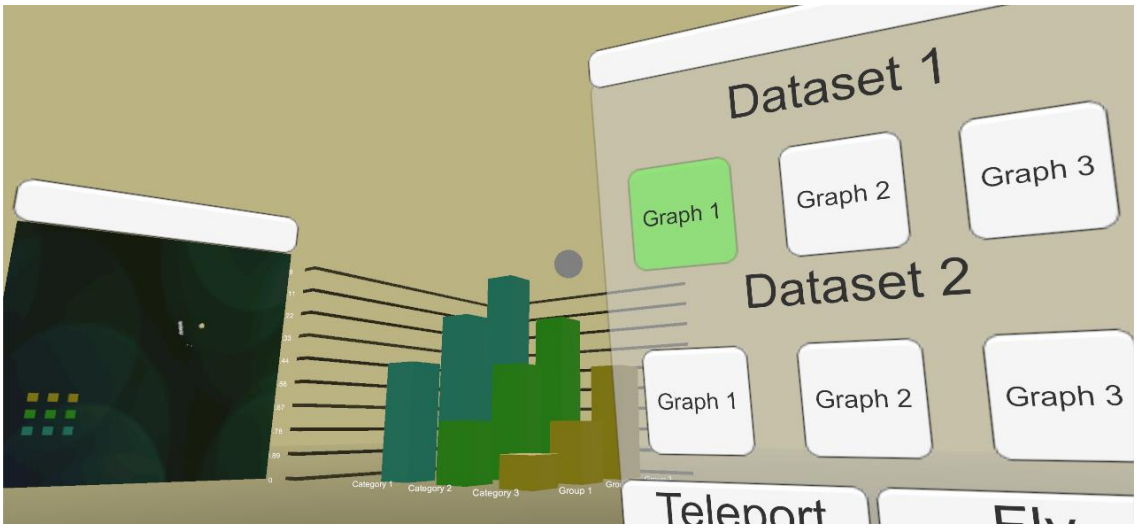

Figura 29. Mudança de cena pelo menu, em acção.

10. **Modificar técnicas visuais para exibir informações 3D discretas;**

Este ponto aplica-se apenas à técnica gráfico de bolhas, pois o gráfico de barras está funcional. Os gráficos de bolhas possíveis de criar com o *asset Graph and Chart* estão feitos para mostrar a mesma informação tanto em 2D quanto em 3D mudando apenas o estilo. No entanto, ao gerar as categorias através de um script, conseguimos mostrar informação 3D discreta.

O seguinte código cria categorias com 31 dias e adiciona uma distância entre elas (*depth*).

```
string[] days = { "1", "2", "3", "4", "5", "6", "7", "8", "9", "10",
"11", "12", "13", "14", "15", "16", "17", "18", "19", "20", "21", "22",
"23", "24", "25", "26", "27", "28", "29", "30", "31" };

for (var j = 0; j < 31; j++)
{
      var lineTiling = new MaterialTiling(true, 20);
```

```
      var depth = j + 2;

      graphChart.DataSource.AddCategory3DGraph(days[j], linePrefab,
lineMaterial, lineThickness, lineTiling, fillPrefab, fillMaterial,
stretchFill, pointPrefab, pointMaterial, pointSize, depth, isCurve,
SegmentsPerCurve);

}
```

Posteriormente, para adicionar pontos ao gráfico

```
graphChart.DataSource.AddPointToCategory(day_string, x, y, pointSize);
```

Assim conseguimos controlar a informação em três locais.

XIII. *Data-Room* (código)

GazeSystem.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class GazeSystem : MonoBehaviour {

    public GameObject reticle;

    public Color inactiveReticleColor = Color.gray;

    public Color activeReticleColor = Color.green;

    private GazableObject currentGazeObject;

    private GazableObject currentSelectedObject;

    private RaycastHit lastHit;

    // Use this for initialization
    void Start ()
    {
        SetReticleColor(inactiveReticleColor);
      }

      // Update is called once per frame
      void Update ()
    {
        ProcessGaze();
        CheckForInput(lastHit);
      }

    public void ProcessGaze()
    {
        Ray raycastRay = new Ray(transform.position, transform.forward);
        RaycastHit hitInfo;

        Debug.DrawRay(raycastRay.origin, raycastRay.direction * 100);

        if(Physics.Raycast(raycastRay, out hitInfo))
        {

            //Do something to the object

            //Check if the object is interactable

            //Get the gameobject from the hitInfo
            GameObject hitObj = hitInfo.collider.gameObject;

            //Get the GazableObject from the hit object
            GazableObject gazeObj =
hitObj.GetComponentInParent<GazableObject>();

            //Object has a Gazeable component
            if (gazeObj != null)
            {
```

```
                //Object we're looking at is different
                if(gazeObj != currentGazeObject)
                {

                    ClearCurrentObject();
                    currentGazeObject = gazeObj;
                    currentGazeObject.OnGazeEnter(hitInfo);
                    SetReticleColor(activeReticleColor);
                }
                else
                {
                    currentGazeObject.OnGaze(hitInfo);
                }
            }
            else
            {
                ClearCurrentObject();
            }

            lastHit = hitInfo;

        }
        else
        {
            ClearCurrentObject();
        }
    }

    private void SetReticleColor(Color reticleColor)
    {
        //Set the color of the reticle
        reticle.GetComponent<Renderer>().material.SetColor("_Color",
reticleColor);
    }

    private void CheckForInput(RaycastHit hitInfo)
    {
        //Check for down
        if(Input.GetMouseButtonDown(0) && currentGazeObject != null)
        {
            currentSelectedObject = currentGazeObject;
            currentSelectedObject.OnPress(hitInfo);

        }

        //Check for hold
        else if (Input.GetMouseButton(0) && currentSelectedObject !=
null)
        {
            currentSelectedObject.OnHold(hitInfo);

        }
        //Check for release
        else if (Input.GetMouseButtonUp(0) && currentSelectedObject !=
null)
        {
            currentSelectedObject.OnRelease(hitInfo);
            currentSelectedObject = null;
        }
```

```
    }

    private void ClearCurrentObject()
    {

        if(currentGazeObject != null)
        {
            currentGazeObject.OnGazeExit();
            SetReticleColor(inactiveReticleColor);
            currentGazeObject = null;
        }

    }

}
```

GazeableObject.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class GazableObject : MonoBehaviour
{
    private const int IGNORE_RAYCAST_LAYER = 2;

    private void Start()
    {

    }

    public virtual void OnGazeEnter(RaycastHit hitInfo)
    {
        Debug.Log("Gaze entered on" + gameObject.name);

    }

    public virtual void OnGaze(RaycastHit hitInfo)
    {
        Debug.Log("Gaze hold on" + gameObject.name);
    }

    public virtual void OnGazeExit()
    {
        Debug.Log("Gaze exited on" + gameObject.name);

    }

    public virtual void OnPress(RaycastHit hitInfo)
    {
        Debug.Log("Button pressed");

    }

    public virtual void OnHold(RaycastHit hitInfo)
    {
        Debug.Log("Button hold");
```

```
    }

    public virtual void OnRelease(RaycastHit hitInfo)
    {
        Debug.Log("Button released");

    }

    public virtual void GazeTranslate(RaycastHit hitInfo)
    {
        //Move the object (position)
        if(hitInfo.collider != null &&
hitInfo.collider.GetComponent<Floor>())
        {
            transform.position = hitInfo.point;
        }
    }
}
```

GazeableButton.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

[RequireComponent(typeof(Image))]
public class GazableButton : GazableObject {

    protected VRCanvas parentPanel;

      // Use this for initialization
      void Start () {
        parentPanel = GetComponentInParent<VRCanvas>();
      }

      // Update is called once per frame
      void Update () {

      }

    public void SetButtonColor(Color buttonColor)
    {
        GetComponent<Image>().color = buttonColor;

    }

    public override void OnPress(RaycastHit hitInfo)
    {
        base.OnPress(hitInfo);
        if(parentPanel != null)
        {
            parentPanel.SetActiveButton(this);
        }
        else
        {
```

```
            Debug.LogError("Button not a child of object with VRPanel
component.", this);
        }
    }

}
```

VRCanvas.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class VRCanvas : MonoBehaviour {

    public GazableButton currentActiveButton;

    public Color unselectedColor = Color.white;
    public Color selectedColor = Color.green;

    // Use this for initialization
    void Start () {

      }

      // Update is called once per frame
      void Update () {
        LookAtPlayer();
      }

    public void SetActiveButton(GazableButton activeButton)
    {
        if(currentActiveButton != null)
        {
            currentActiveButton.SetButtonColor(unselectedColor);
        }

        if(activeButton !=null && currentActiveButton != activeButton)
        {
            currentActiveButton = activeButton;
            currentActiveButton.SetButtonColor(selectedColor);
        }
        else
        {
            Debug.Log("Resetting");
            currentActiveButton = null;
            Player.instance.activeMode = InputMode.NONE;
        }

    }

    public void LookAtPlayer()
    {
        Vector3 playerPos = Player.instance.transform.position;
        Vector3 vecToPlayer = playerPos - transform.position;

        Vector3 lookAtPos = transform.position - vecToPlayer;
```

```
        transform.LookAt(lookAtPos);
    }

}
```

Player.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;

public enum InputMode
{
    NONE,
    TELEPORT,
    FLY,
    DRAG,
    Vis1,
    Vis2,
    Vis3,
    Vis4,
    Vis5,
    Vis6
}

public class Player : MonoBehaviour {

    public static Player instance; //Singleton design pattern: only one
instance of the class appears
    public InputMode activeMode = InputMode.NONE;

    [SerializeField]
    private float playerSpeed = 3.0f;

    void Awake()
    {
        if(instance != null)
        {
            GameObject.Destroy(instance.gameObject);
        }
        instance = this;
    }

    // Use this for initialization
    void Start () {

      }

      // Update is called once per frame
      void Update ()
    {
        TryFly();
        GoTo1();
        GoTo2();
        GoTo3();
```

```
        GoTo4();
        GoTo5();
        GoTo6();
    }

    public void GoTo1()
    {
        if (activeMode == InputMode.Vis1)
        {
            SceneManager.LoadScene("1");

        }
    }

    public void GoTo2()
    {
        if (activeMode == InputMode.Vis2)
        {
            SceneManager.LoadScene("2");

        }
    }

    public void GoTo3()
    {
        if (activeMode == InputMode.Vis3)
        {
            SceneManager.LoadScene("3");

        }
    }
    public void GoTo4()
    {
        if (activeMode == InputMode.Vis4)
        {
            SceneManager.LoadScene("4");

        }
    }
    public void GoTo5()
    {
        if (activeMode == InputMode.Vis5)
        {
            SceneManager.LoadScene("5");

        }
    }
    public void GoTo6()
    {
        if (activeMode == InputMode.Vis6)
        {
            SceneManager.LoadScene("6");

        }
    }

    public void TryFly()
    {
        if(Input.GetMouseButton(0) && activeMode == InputMode.FLY)
```

```
        {
            Vector3 forward = Camera.main.transform.forward;

            Vector3 newPosition = transform.position + forward *
Time.deltaTime * playerSpeed;

            transform.position = newPosition;
        }
    }
}
```

ModeButton.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class ModeButton : GazableButton {

    [SerializeField]
    private InputMode mode;

    public override void OnPress(RaycastHit hitInfo)
    {
        base.OnPress(hitInfo);

        if(parentPanel.currentActiveButton != null)
        {
            Player.instance.activeMode = mode;
        }
    }
}
```

Floor.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Floor : GazableObject {

    public override void OnPress(RaycastHit hitInfo)
    {
        base.OnPress(hitInfo);

        if(Player.instance.activeMode == InputMode.TELEPORT)
        {
            Vector3 destLocation = hitInfo.point;

            destLocation.y = Player.instance.transform.position.y;

            Player.instance.transform.position = destLocation;

        }
    }
}
```

DragZone.cs

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class DragZone : GazableObject {

    private VRCanvas parentPanel;

    private Transform originalParent;

    private InputMode savedInputMode = InputMode.NONE;

      // Use this for initialization
      void Start () {
        parentPanel = GetComponentInParent<VRCanvas>();
      }

      // Update is called once per frame
      void Update () {

      }

    public override void OnPress(RaycastHit hitInfo)
    {
        base.OnPress(hitInfo);

        //Make the entire canvas a child of the camera to move with it.
        originalParent = parentPanel.transform.parent;
        parentPanel.transform.parent = Camera.main.transform;

        //Save the old input mode and set the current mode to drag
        savedInputMode = Player.instance.activeMode;
        Player.instance.activeMode = InputMode.DRAG;
    }

    public override void OnRelease(RaycastHit hitInfo)
    {
        base.OnRelease(hitInfo);

        //Reapply the old values
        parentPanel.transform.parent = originalParent;
        Player.instance.activeMode = savedInputMode;
    }

}
```

XIV. Instalar R e RStudio

Se desejamos instalar R para trabalhar no nosso próprio computador, podemos fazer download gratuitamente no Comprehensive R Archive Network (CRAN) https://cran.r-project.org/ (acedido dia 16 de abril de 2019). De notar que o CRAN disponibiliza várias versões do R: versões para vários sistemas operativos e versões anteriores à atual. É aconselhável ler e seguir as instruções do CRAN para assegurar que foi instalada a versão correta.

Iremos explicar apenas como fazê-lo para o Windows.

1. Abrir o browser e navegar para https://cran.r-project.org/ (acedido dia 16 de abril de 2019).
2. Clicar em "Download R for Windows".
3. Clicar em "base".
4. Clicar em "Download R 3.5.3. for Windows" (Figura 30).

Figura 30. Download R 3.5.3 para Windows.

5. Abrir e correr o ficheiro R-3.5.3-win.exe. Não é necessário alterar o *default* da instalação (Figura 31).

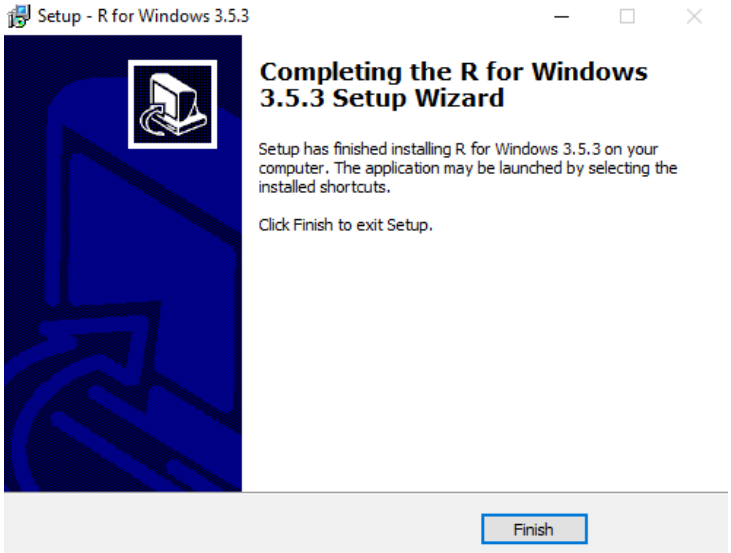

Figura 31. Instalação de R completa.

O RStudio é um ambiente de desenvolvimento integrado (IDE). É altamente recomendável instalar e usar o RStudio para editar e testar o código. Podemos instalar o RStudio no site do RStudio https://www.rstudio.com/products/rstudio/download/ (acedido dia 16 de abril de 2019). No entanto, precisamos de instalar o R primeiro.

Iremos explicar apenas como fazê-lo para o Windows.

1. Abrir o browser, navegar para https://www.rstudio.com/ (acedido dia 16 de abril de 2019) e clicar em "Download RStudio".

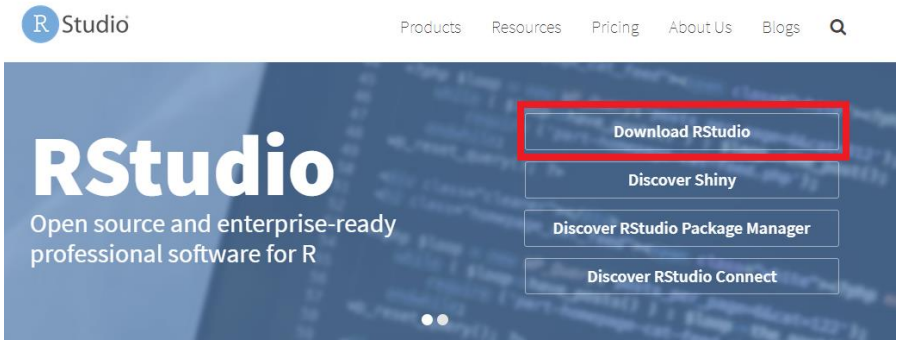

Figura 32. Download RStudio.

2. Clicar no botão "DOWNLOAD" em RStudio Desktop.

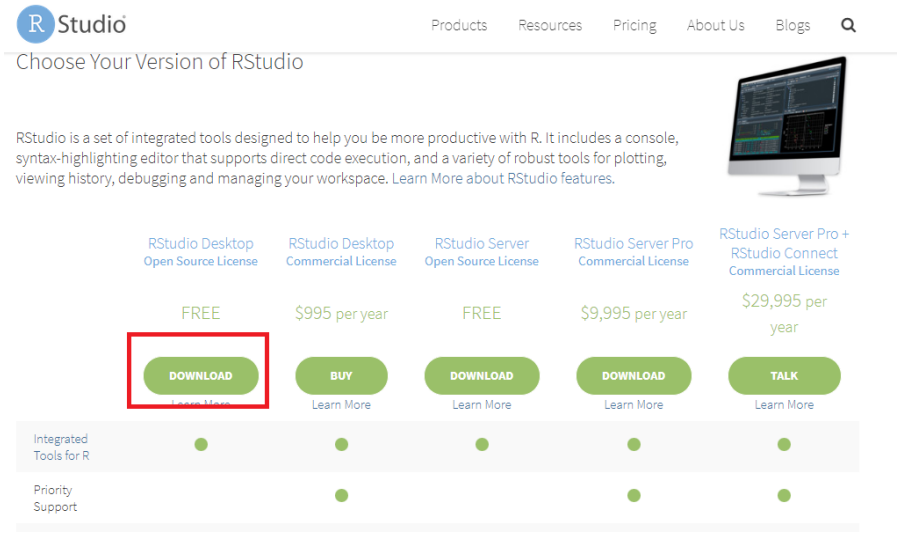

Figura 33. Download RStudio Desktop.

3. Fazer download do RStudio 1.2.1335 – Windows 7+.

Installers for Supported Platforms

Installers	Size	Date	MD5
RStudio 1.2.1335 - Windows 7+	126.9 MB	2019-04-08	d0e2470f1f8ef4cd35a669aa323a2136
RStudio 1.2.1335 - Mac OS X 10.12+ (64-bit)	121.1 MB	2019-04-08	6c570b0e2144583f7c48c284ce299eef
RStudio 1.2.1335 - Ubuntu 14/Debian 8 (64-bit)	92.2 MB	2019-04-08	c1b07d0511469abfe582919b183eee83
RStudio 1.2.1335 - Ubuntu 16 (64-bit)	99.3 MB	2019-04-08	c142d69c210257fb10d18c045fff13c7
RStudio 1.2.1335 - Ubuntu 18 (64-bit)	100.4 MB	2019-04-08	71a8d1990c0d97939804b46cfb0aea75
RStudio 1.2.1335 - Fedora 19+/RedHat 7+ (64-bit)	114.1 MB	2019-04-08	296b6ef88969a91297fab6545f256a7a
RStudio 1.2.1335 - Debian 9+ (64-bit)	100.6 MB	2019-04-08	1e32d4d6f6e216f086a81ca82ef65a91
RStudio 1.2.1335 - OpenSUSE 15+ (64-bit)	101.6 MB	2019-04-08	2795a63c7efd8e2aa2dae86ba09a81e5
RStudio 1.2.1335 - SLES/OpenSUSE 12+ (64-bit)	94.4 MB	2019-04-08	c65424b06ef6737279d982db9eefcae1

Figura 34. Download Rstudio – Windows 7+.

4. Abrir e correr o ficheiro RStudio- 1.2.1335.exe. Não é necessário alterar o *default* da instalação.

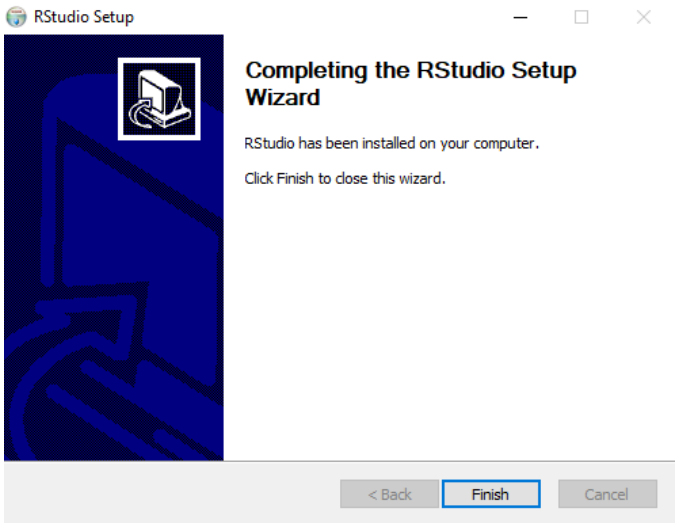

Figura 35. Instalação de RStudio completa.

XV. Instalar Jupiter Notebook num Python virtualenv

Quem está a ler isto, quer experimentar criar visualizações iguais ou semelhantes e é principiante, sugiro que faça download do Anaconda, como explico neste link https://qr.ae/TWFAUs, iniciar o Spyder (ADI de Python) e utilizar o código colocado no *Jupyter Notebooks*.

Quanto ao procedimento seguido, iremos explicar apenas como fazê-lo para o Windows.

1. Aceder ao PowerShell como administrador.
2. Criar pasta na raiz da raiz

```
PS > cd C:\
PS > mkdir InfoVis
PS > cd InfoVis
```

3. Adquirir Python 3.7.5 (32 bits) e instalar no PATH – (https://www.python.org/downloads/release/python-375/)
   Para quem já tem Python instalado e quer fazer *upgrade* pode simplesmente remover a versão antiga do PATH, fazer download da versão que pretende e adicionar ao PATH.

4. Adquirir pip. Para isso basta abrir o seguinte ficheiro (https://bootstrap.pypa.io/get-pip.py) e guardar como *get-pip.py*.

```
PS c:\InfoVis> python get-pip.py
```

Para averiguar se está tudo funcional,

```
PS C:\InfoVis> pip
PS C:\InfoVis> pip install –upgrade setuptools
PS C:\InfoVis> pip install ez_setup
```

5. Instalar virtualenv e virtualenvwrapper-powershell

```
PS C:\InfoVis> pip install virtualenv
PS C:\InfoVis> pip install virtualenvwrapper-powershell
```

6. Criar e ativar ambiente virtual

```
PS C:\InfoVis\virtualenvs> virtualenv masters
PS C:\InfoVis\virtualenv> Set-ExecutionPolicy Unrestricted -Force
PS C:\InfoVis\virtualenv> masters\Scripts\activate
(dev) PS C:\InfoVis\virtualenvs\ >
```

7. Instalar Python Mayavi

```
(dev) PS C:\InfoVis\virtualenvs\dev> pip install mayavi
```

8. Instalar jupyterlab

```
(dev) PS C:\InfoVis\virtualenvs\dev> pip install jupyterlab
```

9. Instalar PyQt5

```
(dev) PS C:\InfoVis\virtualenvs\dev> pip install PyQt5
```

10. Iniciar o Jupyter Notebook

```
(dev) PS C:\InfoVis\virtualenvs\dev> jupyter notebook
```

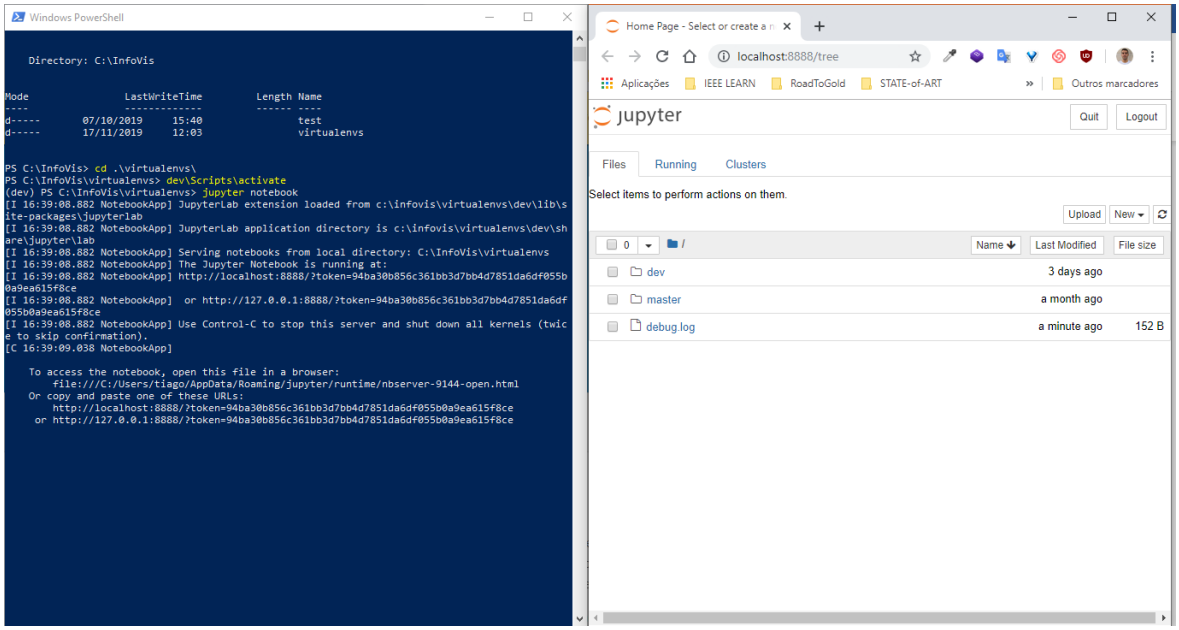

Figura 36. Jupyter Notebook dentro de um Python virtualenv.

XVI. Extração e tratamento dos dados

Para podermos extrair todos os dados resultantes da *query* para o nosso ambiente de trabalho, configurámos a *query* para que a mesma não tenha limites (*allowLargeResults*) e armazenamos os resultados numa tabela. Posteriormente, exportámos a tabela para um *bucket* no Google Cloud Storage e fizemos download dos ficheiros dentro do *bucket* para a nossa máquina. Apenas na primeira *query* tivemos de recorrer a um *wildcard* para exportar para o *bucket* no GCS porque a mesma tinha dimensões superiores a 1 GB.

Em maior detalhe: https://stackoverflow.com/a/58132961/5675325

1.ª CODIFICAÇÃO VISUAL

```
SELECT
  year, mo AS month, ((temp-32)*5/9) AS temperature
FROM
  `bigquery-public-data.noaa_gsod.gsod*`
```

2.ª CODIFICAÇÃO VISUAL

```
SELECT
  year, mo AS month, ((temp-32)*5/9) AS temperature
FROM
  `bigquery-public-data.noaa_gsod.gsod*`
WHERE
  CAST(YEAR AS INT64) > 2008
ORDER BY year
```

3.ª CODIFICAÇÃO VISUAL

```
SELECT
  year, mo AS month, ((temp-32)*5/9) AS temperature
FROM
  `bigquery-public-data.noaa_gsod.gsod*`
WHERE
  CAST(YEAR AS INT64) > 2008
  AND (stn="085300")    -- CABO CARVOEIRO
ORDER BY year
```

4.ª CODIFICAÇÃO VISUAL

```
SELECT
  year, da AS day, ((temp-32)*5/9) AS temperature
FROM
  `bigquery-public-data.noaa_gsod.gsod*`
WHERE
  (stn="085300")    -- CABO CARVOEIRO
ORDER BY year
```

5.ª CODIFICAÇÃO VISUAL

```
SELECT
  year, da AS day, ((temp-32)*5/9) AS temperature
FROM
  `bigquery-public-data.noaa_gsod.gsod*`
WHERE
  CAST(YEAR AS INT64) > 2008
  AND (stn="085300")    -- CABO CARVOEIRO
ORDER BY year
```

6.ª CODIFICAÇÃO VISUAL

```
SELECT
  mo AS month, da AS day, ((temp-32)*5/9) AS temperature
FROM
  `bigquery-public-data.noaa_gsod.gsod*`
WHERE
  CAST(YEAR AS INT64) > 2008
  AND (stn="085300")    -- CABO CARVOEIRO
ORDER BY month
```

De notar que na primeira *query* não temos no comando SQL "ORDER BY" simplesmente porque o BigQuery levanta um erro indicando que os recursos excedem a execução de consulta (*Resources Exceeds Query Execution*) demonstrando que, quando permitimos resultados grandes, chega a um ponto em que a *query* não pode mais ser calculada em paralelo.

Além disso, nesse caso fomos forçados a extrair múltiplos CSVs do *bucket* e precisámos de juntá-los localmente (recorremos ao *script* em Python mencionado em Anexos, secção VIII). Para cada ambiente utilizámos CSVs com diferentes números de linhas, consoante. Para criar esses CSVs recorremos ao script presente em Anexos, secção IX. Para situações em que a ordem é relevante, ou seja, os dados precisam de estar ordenados, ver Anexos, secção XI.

XVII. Visualizações com Data Room

1.ª CODIFICAÇÃO VISUAL

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using ChartAndGraph;
using System;

public class BarSample1 : MonoBehaviour
{

    Material categoryMaterial;
    Color hoverColor, selectedColor;
    public BarChart barChart;
    public Material newMaterialRef;

    void Awake()
    {
        //BarChart barChart = GetComponent<BarChart>();

        //if (barChart != null)
        //{
        barChart.DataSource.ClearCategories();
        barChart.DataSource.ClearGroups();

        List<List<object>> data = CSVReaderNoHeader.Read("1");
        //Debug.Log(data);
        print(data);

        string[] months = { "1", "2", "3", "4", "5", "6", "7", "8", "9",
"10", "11", "12" };
        for (var j = 0; j < 12; j++)
        {
            barChart.DataSource.AddCategory(months[j], newMaterialRef);
        }


        for (var i = 0; i < data.Count; i++)
        //for (var i = 0; i < 100; i++)
        {

            //double? month = data[i][0] as double?;
            //if (!month.HasValue)
            //{
            //    print($"Month value on line {i + 1} is corrupted;
Skipping...");
            //    break;
            //}
            double month = (double)(int)data[i][1];
            string month_string = month.ToString();

            print($"month:{month} ///////END");
            print($"month_string:{month_string} ///////END");

            //double? year = data[i][1] as double?;
            //if (!year.HasValue)
            //{
```

```
            //    print($"Year value on line {i + 1} is corrupted;
Skipping...");
            //    break;
            //}
            double year = (double)(int)data[i][0];
            string year_string = year.ToString();

            print($"year:{year} ///////END");
            print($"year_string:{year_string} ///////END");

            //double? temperature = data[i][2] as double?;
            //if (!temperature.HasValue)
            //{
            //    print($"Temperature value on line {i + 1} is corrupted;
Skipping...");
            //    break;
            //}
            Console.Write(data[i][2].GetType());

            double temperature;
            double.TryParse(data[i][2].ToString(),
System.Globalization.NumberStyles.Any,
System.Globalization.CultureInfo.InvariantCulture, out temperature);
            // Another way: double temperature =
System.Xml.XmlConvert.ToDouble(data[i][2].ToString());
            print($"temperature:{temperature} ///////END");

            if (barChart.DataSource.HasCategory(month_string) &&
barChart.DataSource.HasGroup(year_string))
            {
                barChart.DataSource.SetValue(month_string, year_string,
temperature);
            }
            //if (!barChart.DataSource.HasCategory(month_string) &&
barChart.DataSource.HasGroup(year_string))
            //{
            //    barChart.DataSource.AddCategory(month_string, new
ChartDynamicMaterial(categoryMaterial, hoverColor, selectedColor));
            //    barChart.DataSource.SetValue(month_string, year_string,
temperature);
            //}
            if (barChart.DataSource.HasCategory(month_string)
&& !barChart.DataSource.HasGroup(year_string))
            {
                barChart.DataSource.AddGroup(year_string);
                barChart.DataSource.SetValue(month_string, year_string,
temperature);
            }
            //if (!barChart.DataSource.HasCategory(month_string)
&& !barChart.DataSource.HasGroup(year_string))
            //{
            //    barChart.DataSource.AddCategory(month_string, new
ChartDynamicMaterial(categoryMaterial, hoverColor, selectedColor));
            //    barChart.DataSource.AddGroup(year_string);
            //    barChart.DataSource.SetValue(month_string, year_string,
temperature);
            //}

        }
```

```
    }

    //Update is called once per frame
    void Update()
    {

    }
}
```

2.ª CODIFICAÇÃO VISUAL

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using ChartAndGraph;
using System;

public class BarSample2 : MonoBehaviour
{

    Material categoryMaterial;
    Color hoverColor, selectedColor;
    public BarChart barChart;
    public Material newMaterialRef;

    void Awake()
    {
        //BarChart barChart = GetComponent<BarChart>();
        //if (barChart != null)
        //{
        barChart.DataSource.ClearCategories();
        barChart.DataSource.ClearGroups();

        List<List<object>> data = CSVReaderNoHeader.Read("2");
        //Debug.Log(data);
        print(data);

        string[] months = { "1", "2", "3", "4", "5", "6", "7", "8", "9",
"10", "11", "12" };
        for (var j = 0; j < 12; j++)
        {
            barChart.DataSource.AddCategory(months[j], newMaterialRef);
        }


        for (var i = 0; i < data.Count; i++)
        //for (var i = 0; i < 100; i++)
        {

            //double? month = data[i][0] as double?;
            //if (!month.HasValue)
            //{
            //    print($"Month value on line {i + 1} is corrupted;
Skipping...");
            //    break;
            //}
```

```
            double month = (double)(int)data[i][1];
            string month_string = month.ToString();

            print($"month:{month} ///////END");
            print($"month_string:{month_string} ///////END");

            //double? year = data[i][1] as double?;
            //if (!year.HasValue)
            //{
            //    print($"Year value on line {i + 1} is corrupted;
Skipping...");
            //    break;
            //}
            double year = (double)(int)data[i][0];
            string year_string = year.ToString();

            print($"year:{year} ///////END");
            print($"year_string:{year_string} ///////END");

            //double? temperature = data[i][2] as double?;
            //if (!temperature.HasValue)
            //{
            //    print($"Temperature value on line {i + 1} is corrupted;
Skipping...");
            //    break;
            //}
            Console.Write(data[i][2].GetType());

            double temperature;
            double.TryParse(data[i][2].ToString(),
System.Globalization.NumberStyles.Any,
System.Globalization.CultureInfo.InvariantCulture, out temperature);
            // Another way: double temperature =
System.Xml.XmlConvert.ToDouble(data[i][2].ToString());
            print($"temperature:{temperature} ///////END");

            if (barChart.DataSource.HasCategory(month_string) &&
barChart.DataSource.HasGroup(year_string))
            {
                barChart.DataSource.SetValue(month_string, year_string,
temperature);
            }
            if (!barChart.DataSource.HasCategory(month_string) &&
barChart.DataSource.HasGroup(year_string))
            {
                barChart.DataSource.AddCategory(month_string, new
ChartDynamicMaterial(categoryMaterial, hoverColor, selectedColor));
                barChart.DataSource.SetValue(month_string, year_string,
temperature);
            }
            if (barChart.DataSource.HasCategory(month_string)
&& !barChart.DataSource.HasGroup(year_string))
            {
                barChart.DataSource.AddGroup(year_string);
                barChart.DataSource.SetValue(month_string, year_string,
temperature);
            }
            if (!barChart.DataSource.HasCategory(month_string)
&& !barChart.DataSource.HasGroup(year_string))
```

```
            {
                barChart.DataSource.AddCategory(month_string, new
ChartDynamicMaterial(categoryMaterial, hoverColor, selectedColor));
                barChart.DataSource.AddGroup(year_string);
                barChart.DataSource.SetValue(month_string, year_string,
temperature);
            }

        }
    }

    //Update is called once per frame
    void Update()
    {

    }
}
```

3.ª CODIFICAÇÃO VISUAL

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using ChartAndGraph;
using System;

public class BarSample3 : MonoBehaviour
{

    Material categoryMaterial;
    Color hoverColor, selectedColor;
    public BarChart barChart;
    public Material newMaterialRef;

    void Awake()
    {
        //BarChart barChart = GetComponent<BarChart>();

        //if (barChart != null)
        //{
        barChart.DataSource.ClearCategories();
        barChart.DataSource.ClearGroups();

        List<List<object>> data = CSVReaderNoHeader.Read("3");
        //Debug.Log(data);
        print(data);

        string[] months = { "1", "2", "3", "4", "5", "6", "7", "8", "9",
"10", "11", "12" };
        for (var j = 0; j < 12; j++)
        {
            barChart.DataSource.AddCategory(months[j], newMaterialRef);
        }


        for (var i = 0; i < data.Count; i++)
        //for (var i = 0; i < 100; i++)
```

```
        {

            //double? month = data[i][0] as double?;
            //if (!month.HasValue)
            //{
            //    print($"Month value on line {i + 1} is corrupted;
Skipping...");
            //    break;
            //}
            double month = (double)(int)data[i][1];
            string month_string = month.ToString();

            print($"month:{month} ///////END");
            print($"month_string:{month_string} ///////END");

            //double? year = data[i][1] as double?;
            //if (!year.HasValue)
            //{
            //    print($"Year value on line {i + 1} is corrupted;
Skipping...");
            //    break;
            //}
            double year = (double)(int)data[i][0];
            string year_string = year.ToString();

            print($"year:{year} ///////END");
            print($"year_string:{year_string} ///////END");

            //double? temperature = data[i][2] as double?;
            //if (!temperature.HasValue)
            //{
            //    print($"Temperature value on line {i + 1} is corrupted;
Skipping...");
            //    break;
            //}
            Console.Write(data[i][2].GetType());

            double temperature;
            double.TryParse(data[i][2].ToString(),
System.Globalization.NumberStyles.Any,
System.Globalization.CultureInfo.InvariantCulture, out temperature);
            // Another way: double temperature =
System.Xml.XmlConvert.ToDouble(data[i][2].ToString());
            print($"temperature:{temperature} ///////END");

            if (barChart.DataSource.HasCategory(month_string) &&
barChart.DataSource.HasGroup(year_string))
            {
                barChart.DataSource.SetValue(month_string, year_string,
temperature);
            }
            //if (!barChart.DataSource.HasCategory(month_string) &&
barChart.DataSource.HasGroup(year_string))
            //{
            //    barChart.DataSource.AddCategory(month_string, new
ChartDynamicMaterial(categoryMaterial, hoverColor, selectedColor));
            //    barChart.DataSource.SetValue(month_string, year_string,
temperature);
            //}
```

```
            if (barChart.DataSource.HasCategory(month_string)
&& !barChart.DataSource.HasGroup(year_string))
            {
                barChart.DataSource.AddGroup(year_string);
                barChart.DataSource.SetValue(month_string, year_string,
temperature);
            }
            //if (!barChart.DataSource.HasCategory(month_string)
&& !barChart.DataSource.HasGroup(year_string))
            //{
            //    barChart.DataSource.AddCategory(month_string, new
ChartDynamicMaterial(categoryMaterial, hoverColor, selectedColor));
            //    barChart.DataSource.AddGroup(year_string);
            //    barChart.DataSource.SetValue(month_string, year_string,
temperature);
            //}

        }


    }

    //Update is called once per frame
    void Update()
    {

    }
}
```

4.ª CODIFICAÇÃO VISUAL

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using ChartAndGraph;
using System;

public class GraphSample1 : MonoBehaviour
{

    public GraphChartBase graphChart;
    public Material lineMaterial, pointMaterial, fillMaterial;
    public PathGenerator linePrefab;
    public FillPathGenerator pathPrefab, fillPrefab;
    public GameObject pointPrefab;
    public double lineThickness = 2.0, pointSize = 5.0;
    public bool stretchFill = false;
    public bool isCurve = false;
    public int SegmentsPerCurve;


    // Start is called before the first frame update
    void Start()
    {
        List<List<object>> data = CSVReaderNoHeader.Read("4");
        //Debug.Log(data);
```

```
        print(data);

        string[] days = { "1", "2", "3", "4", "5", "6", "7", "8", "9",
"10", "11", "12", "13", "14", "15", "16", "17", "18", "19", "20", "21",
"22", "23", "24", "25", "26", "27", "28", "29", "30", "31" };
        for (var j = 0; j < 31; j++)
        {
            var lineTiling = new MaterialTiling(true, 20); // see the
article about material tiling
            var depth = j + 2;

            graphChart.DataSource.AddCategory3DGraph(days[j], linePrefab,
lineMaterial, lineThickness, lineTiling, fillPrefab, fillMaterial,
stretchFill, pointPrefab, pointMaterial, pointSize, depth, isCurve,
SegmentsPerCurve);

        }

        for (var i = 0; i < data.Count; i++)
        //for (var i = 0; i < 100; i++)
        {

            //double? month = data[i][0] as double?;
            //if (!month.HasValue)
            //{
            //    print($"Month value on line {i + 1} is corrupted;
Skipping...");
            //    break;
            //}
            double day = (double)(int)data[i][1];
            string day_string = day.ToString();

            print($"day:{day} ///////END");
            print($"month_string:{day_string} ///////END");

            //double? year = data[i][1] as double?;
            //if (!year.HasValue)
            //{
            //    print($"Year value on line {i + 1} is corrupted;
Skipping...");
            //    break;
            //}
            double year = (double)(int)data[i][0];
            string year_string = year.ToString();

            print($"year:{year} ///////END");
            print($"year_string:{year_string} ///////END");

            //double? temperature = data[i][2] as double?;
            //if (!temperature.HasValue)
            //{
            //    print($"Temperature value on line {i + 1} is corrupted;
Skipping...");
            //    break;
            //}
            Console.Write(data[i][2].GetType());

            double temperature;
```

```
            double.TryParse(data[i][2].ToString(),
System.Globalization.NumberStyles.Any,
System.Globalization.CultureInfo.InvariantCulture, out temperature);
            // Another way: double temperature =
System.Xml.XmlConvert.ToDouble(data[i][2].ToString());
            print($"temperature:{temperature} ///////END");

            if (graphChart.DataSource.HasCategory(day_string))
            {
                graphChart.DataSource.AddPointToCategory(day_string,
year, temperature, temperature);

            }


        }

    }

    // Update is called once per frame
    void Update()
    {

    }
}
```

5.ª CODIFICAÇÃO VISUAL

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using ChartAndGraph;
using System;

public class GraphSample2 : MonoBehaviour
{

    public GraphChartBase graphChart;
    public Material lineMaterial, pointMaterial, fillMaterial;
    public PathGenerator linePrefab;
    public FillPathGenerator pathPrefab, fillPrefab;
    public GameObject pointPrefab;
    public double lineThickness = 2.0, pointSize = 5.0;
    public bool stretchFill = false;
    public bool isCurve = false;
    public int SegmentsPerCurve;


    // Start is called before the first frame update
    void Start()
    {
        List<List<object>> data = CSVReaderNoHeader.Read("5");
        //Debug.Log(data);
        print(data);
```

```
        string[] days = { "1", "2", "3", "4", "5", "6", "7", "8", "9",
"10", "11", "12", "13", "14", "15", "16", "17", "18", "19", "20", "21",
"22", "23", "24", "25", "26", "27", "28", "29", "30", "31" };
        for (var j = 0; j < 31; j++)
        {
            var lineTiling = new MaterialTiling(true, 20); // see the
article about material tiling
            var depth = j + 2;

            graphChart.DataSource.AddCategory3DGraph(days[j], linePrefab,
lineMaterial, lineThickness, lineTiling, fillPrefab, fillMaterial,
stretchFill, pointPrefab, pointMaterial, pointSize, depth, isCurve,
SegmentsPerCurve);

        }

        graphChart.DataSource.AddPointToCategory("0", 2005, 0);
        graphChart.DataSource.AddPointToCategory("0", 2006, 0);
        graphChart.DataSource.AddPointToCategory("0", 2007, 0);
        graphChart.DataSource.AddPointToCategory("0", 2011, 0);
        graphChart.DataSource.AddPointToCategory("0", 2012, 0);
        graphChart.DataSource.AddPointToCategory("0", 2013, 0);
        graphChart.DataSource.AddPointToCategory("0", 2014, 0);

        for (var i = 0; i < data.Count; i++)
        //for (var i = 0; i < 100; i++)
        {

            //double? month = data[i][0] as double?;
            //if (!month.HasValue)
            //{
            //    print($"Month value on line {i + 1} is corrupted;
Skipping...");
            //    break;
            //}
            double day = (double)(int)data[i][1];
            string day_string = day.ToString();

            print($"day:{day} ///////END");
            print($"month_string:{day_string} ///////END");

            //double? year = data[i][1] as double?;
            //if (!year.HasValue)
            //{
            //    print($"Year value on line {i + 1} is corrupted;
Skipping...");
            //    break;
            //}
            double year = (double)(int)data[i][0];
            string year_string = year.ToString();

            print($"year:{year} ///////END");
            print($"year_string:{year_string} ///////END");

            //double? temperature = data[i][2] as double?;
            //if (!temperature.HasValue)
            //{
            //    print($"Temperature value on line {i + 1} is corrupted;
Skipping...");
```

```
            //    break;
            //}
            Console.Write(data[i][2].GetType());

            double temperature;
            double.TryParse(data[i][2].ToString(),
System.Globalization.NumberStyles.Any,
System.Globalization.CultureInfo.InvariantCulture, out temperature);
            // Another way: double temperature =
System.Xml.XmlConvert.ToDouble(data[i][2].ToString());
            print($"temperature:{temperature} ///////END");

            if (graphChart.DataSource.HasCategory(day_string))
            {
                graphChart.DataSource.AddPointToCategory(day_string,
year, temperature, temperature);

            }


        }


    }

    // Update is called once per frame
    void Update()
    {

    }
}
```

6.ª CODIFICAÇÃO VISUAL

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using ChartAndGraph;
using System;

public class GraphSample3 : MonoBehaviour
{

    public GraphChartBase graphChart;
    public Material lineMaterial, pointMaterial, fillMaterial;
    public PathGenerator linePrefab;
    public FillPathGenerator pathPrefab, fillPrefab;
    public GameObject pointPrefab;
    public double lineThickness = 2.0, pointSize = 5.0;
    public bool stretchFill = false;
    public bool isCurve = false;
    public int SegmentsPerCurve;


    // Start is called before the first frame update
    void Start()
```

```
    {
        List<List<object>> data = CSVReaderNoHeader.Read("6");
        //Debug.Log(data);
        print(data);

        string[] days = { "1", "2", "3", "4", "5", "6", "7", "8", "9",
"10", "11", "12", "13", "14", "15", "16", "17", "18", "19", "20", "21",
"22", "23", "24", "25", "26", "27", "28", "29", "30", "31" };
        for (var j = 0; j < 31; j++)
        {
            var lineTiling = new MaterialTiling(true, 20); // see the
article about material tiling
            var depth = j + 2;

            graphChart.DataSource.AddCategory3DGraph(days[j], linePrefab,
lineMaterial, lineThickness, lineTiling, fillPrefab, fillMaterial,
stretchFill, pointPrefab, pointMaterial, pointSize, depth, isCurve,
SegmentsPerCurve);

        }

        //graphChart.DataSource.AddPointToCategory("0", 4, 0);
        //graphChart.DataSource.AddPointToCategory("0", 2006, 0);
        //graphChart.DataSource.AddPointToCategory("0", 2007, 0);
        //graphChart.DataSource.AddPointToCategory("0", 2011, 0);
        //graphChart.DataSource.AddPointToCategory("0", 2012, 0);
        //graphChart.DataSource.AddPointToCategory("0", 2013, 0);
        //graphChart.DataSource.AddPointToCategory("0", 2014, 0);

        for (var i = 0; i < data.Count; i++)
        //for (var i = 0; i < 100; i++)
        {

            //double? month = data[i][0] as double?;
            //if (!month.HasValue)
            //{
            //    print($"Month value on line {i + 1} is corrupted;
Skipping...");
            //    break;
            //}
            double day = (double)(int)data[i][1];
            string day_string = day.ToString();

            print($"day:{day} ///////END");
            print($"month_string:{day_string} ///////END");

            //double? year = data[i][1] as double?;
            //if (!year.HasValue)
            //{
            //    print($"Year value on line {i + 1} is corrupted;
Skipping...");
            //    break;
            //}
            double month = (double)(int)data[i][0];
            string month_string = month.ToString();

            print($"year:{month} ///////END");
            print($"year_string:{month_string} ///////END");
```

```
            //double? temperature = data[i][2] as double?;
            //if (!temperature.HasValue)
            //{
            //    print($"Temperature value on line {i + 1} is corrupted;
Skipping...");
            //    break;
            //}
            Console.Write(data[i][2].GetType());

            double temperature;
            double.TryParse(data[i][2].ToString(),
System.Globalization.NumberStyles.Any,
System.Globalization.CultureInfo.InvariantCulture, out temperature);
            // Another way: double temperature =
System.Xml.XmlConvert.ToDouble(data[i][2].ToString());
            print($"temperature:{temperature} ///////END");

            if (graphChart.DataSource.HasCategory(day_string))
            {
                graphChart.DataSource.AddPointToCategory(day_string,
month, temperature, temperature);

            }


        }


    }

    // Update is called once per frame
    void Update()
    {

    }
}
```

XVIII. Visualizações com Python

1.ª CODIFICAÇÃO VISUAL

```
#!/usr/bin/env python
# coding: utf-8

# In[1]: get_ipython().run_line_magic('gui', 'qt')

# In[2]: from mayavi import mlab

# In[3]: import pandas as pd

# In[4]: df = pd.read_csv('FILELOCATION/FILE.CSV', header=None, sep=',')

# In[5]:df1=df[[0]]
       df2=df[[1]]
       df3=df[[2]]
       mlab.barchart(df1,df2,df3)

# Out[5]: <mayavi.modules.glyph.Glyph at …>

# In[6]:mlab.outline(Out[5])
# In[7]:mlab.colorbar(Out[5], orientation='vertical')
# In[8]:mlab.axes(Out[5])
```

2.ª CODIFICAÇÃO VISUAL

```
#!/usr/bin/env python
# coding: utf-8

# In[1]: get_ipython().run_line_magic('gui', 'qt')

# In[2]: from mayavi import mlab

# In[3]: import pandas as pd

# In[4]: df = pd.read_csv('FILELOCATION/FILE.CSV', header=None, sep=',')

# In[5]:df1=df[[0]]
       df2=df[[1]]
       df3=df[[2]]
       mlab.barchart(df1,df2,df3)

# Out[5]: <mayavi.modules.glyph.Glyph at …>

# In[6]:mlab.outline(Out[5])
# In[7]:mlab.colorbar(Out[5], orientation='vertical')
# In[8]:mlab.axes(Out[5])
```

3.ª CODIFICAÇÃO VISUAL

```
#!/usr/bin/env python
# coding: utf-8

# In[1]: get_ipython().run_line_magic('gui', 'qt')

# In[2]: from mayavi import mlab

# In[3]: import pandas as pd

# In[4]: df = pd.read_csv('FILELOCATION/FILE.CSV', header=None, sep=',')

# In[5]: df1=df[[0]]
       df2=df[[1]]
       df3=df[[2]]
       mlab.barchart(df1,df2,df3)

# Out[5]: <mayavi.modules.glyph.Glyph at …>

# In[6]:mlab.outline(Out[5])
# In[7]:mlab.colorbar(Out[5], orientation='vertical')
# In[8]:mlab.axes(Out[5])
```

4.ª CODIFICAÇÃO VISUAL

```
#!/usr/bin/env python
# coding: utf-8

# In[1]: get_ipython().run_line_magic('gui', 'qt')

# In[2]: from mayavi import mlab

# In[3]: import pandas as pd

# In[4]: df = pd.read_csv('FILELOCATION/FILE.CSV', header=None, sep=',')

# In[5]: df1=df[[0]]
       df2=df[[1]]
       df3=df[[2]]
       mlab.points3d(df1,df2,df3,df3)

# Out[5]: <mayavi.modules.glyph.Glyph at …>

# In[6]:mlab.outline(Out[5])
# In[7]:mlab.colorbar(Out[5], orientation='vertical')
# In[8]:mlab.axes(Out[5])
```

5.ª CODIFICAÇÃO VISUAL

```
#!/usr/bin/env python
# coding: utf-8

# In[1]: get_ipython().run_line_magic('gui', 'qt')

# In[2]: from mayavi import mlab

# In[3]: import pandas as pd

# In[4]: df = pd.read_csv('FILELOCATION/FILE.CSV', header=None, sep=',')

# In[5]:df1=df[[0]]
       df2=df[[1]]
       df3=df[[2]]
       mlab.points3d(df1,df2,df3,df3)

# Out[5]: <mayavi.modules.glyph.Glyph at …>

# In[6]:mlab.outline(Out[5])
# In[7]:mlab.colorbar(Out[5], orientation='vertical')
# In[8]:mlab.axes(Out[5])
```

6.ª CODIFICAÇÃO VISUAL

```
#!/usr/bin/env python
# coding: utf-8

# In[1]: get_ipython().run_line_magic('gui', 'qt')

# In[2]: from mayavi import mlab

# In[3]: import pandas as pd

# In[4]: df = pd.read_csv('FILELOCATION/FILE.CSV', header=None, sep=',')

# In[5]:df1=df[[0]]
       df2=df[[1]]
       df3=df[[2]]
       mlab.points3d(df1,df2,df3,df3)

# Out[5]: <mayavi.modules.glyph.Glyph at …>

# In[6]:mlab.outline(Out[5])
# In[7]:mlab.colorbar(Out[5], orientation='vertical')
# In[8]:mlab.axes(Out[5])
```

XIX. Visualizações com R

1.ª CODIFICAÇÃO VISUAL

```
library(latticeExtra)

heisenberg <- read.csv(file="FILELOCATION/FILE.CSV")

x <- heisenberg[[1]]
y <- heisenberg[[2]]
z <- heisenberg[[3]]

cloud(z~x+y, heisenberg, panel.3d.cloud=panel.3dbars,
      xbase=0.4, ybase=0.4,scales=list(arrows=FALSE, col=1),
      col.facet = level.colors(z, at = do.breaks(range(z), 20),
                               col.regions = terrain.colors,
                               colors = TRUE),
      colorkey = list(col = terrain.colors, at = do.breaks(range(z), 20)),
      par.settings = list(axis.line = list(col = "transparent")))
```

2.ª CODIFICAÇÃO VISUAL

```
library(latticeExtra)

heisenberg <- read.csv(file="FILELOCATION/FILE.CSV")

x <- heisenberg[[1]]
y <- heisenberg[[2]]
z <- heisenberg[[3]]

cloud(z~x+y, heisenberg, panel.3d.cloud=panel.3dbars,
      xbase=0.4, ybase=0.4,scales=list(arrows=FALSE, col=1),
      col.facet = level.colors(z, at = do.breaks(range(z), 20),
                               col.regions = terrain.colors,
                               colors = TRUE),
      colorkey = list(col = terrain.colors, at = do.breaks(range(z), 20)),
      par.settings = list(axis.line = list(col = "transparent")))
```

3.ª CODIFICAÇÃO VISUAL

```
library(latticeExtra)

heisenberg <- read.csv(file="FILELOCATION/FILE.CSV")

x <- heisenberg[[1]]
x <- factor(x, levels = c(2009, 2010), ordered = TRUE)
y <- heisenberg[[2]]
z <- heisenberg[[3]]

cloud(z~x+y, heisenberg, panel.3d.cloud=panel.3dbars,
      xbase=0.4, ybase=0.4,scales=list(arrows=FALSE, col=1),
      col.facet = level.colors(z, at = do.breaks(range(z), 20),
                               col.regions = terrain.colors,
```

```
                                colors = TRUE),
      colorkey = list(col = terrain.colors, at = do.breaks(range(z), 20)),
      par.settings = list(axis.line = list(col = "transparent")))
```

4.ª CODIFICAÇÃO VISUAL

```
library(plot3D)

heisenberg <- read.csv(file="FILELOCATION/FILE.CSV")

x <- heisenberg[[1]]
y <- heisenberg[[2]]
z <- heisenberg[[3]]

points3D(x, y, z, ticktype = "detailed")
```

5.ª CODIFICAÇÃO VISUAL

```
library(plot3D)

heisenberg <- read.csv(file="FILELOCATION/FILE.CSV")

x <- heisenberg[[1]]
y <- heisenberg[[2]]
z <- heisenberg[[3]]

points3D(x, y, z, ticktype = "detailed", xlim =c(2009,2010), nticks = 2)
```

6.ª CODIFICAÇÃO VISUAL

```
library(plot3D)

heisenberg <- read.csv(file="FILELOCATION/FILE.CSV")

x <- heisenberg[[1]]
y <- heisenberg[[2]]
z <- heisenberg[[3]]

points3D(x, y, z, ticktype = "detailed")
```

XX. Estatística descritiva com Python

Aproveitando que já configurámos um ambiente em Python com Jupyter Notebooks na secção XVI, iremos tirar partido desse ambiente nesse momento Este *script* carrega o CSV (sem cabeçalhos) linha por linha na vez de carregar o ficheiro inteiro e, de seguida, mantém dados intermediários no lugar de manter tudo. Por exemplo, para calcular a média, tudo o que precisamos de manter é a soma de todos os valores no lugar de manter todos esses valores. Uma vez alcançando o fim do ficheiro, podemos dividir essa soma através do número de registos para obter média. Para outros (excepto o desvio padrão), é o mesmo.

No caso do desvio padrão, precisámos de ler o ficheiro inteiro duas vezes: uma vez para obter a média (já que não a temos antecipadamente e precisamos de calcular desvios da média) e, na segunda vez, usando a média podemos calcular as diferenças ao quadrado entre todos os valores e a média e, finalmente, fazer a raiz quadrada.

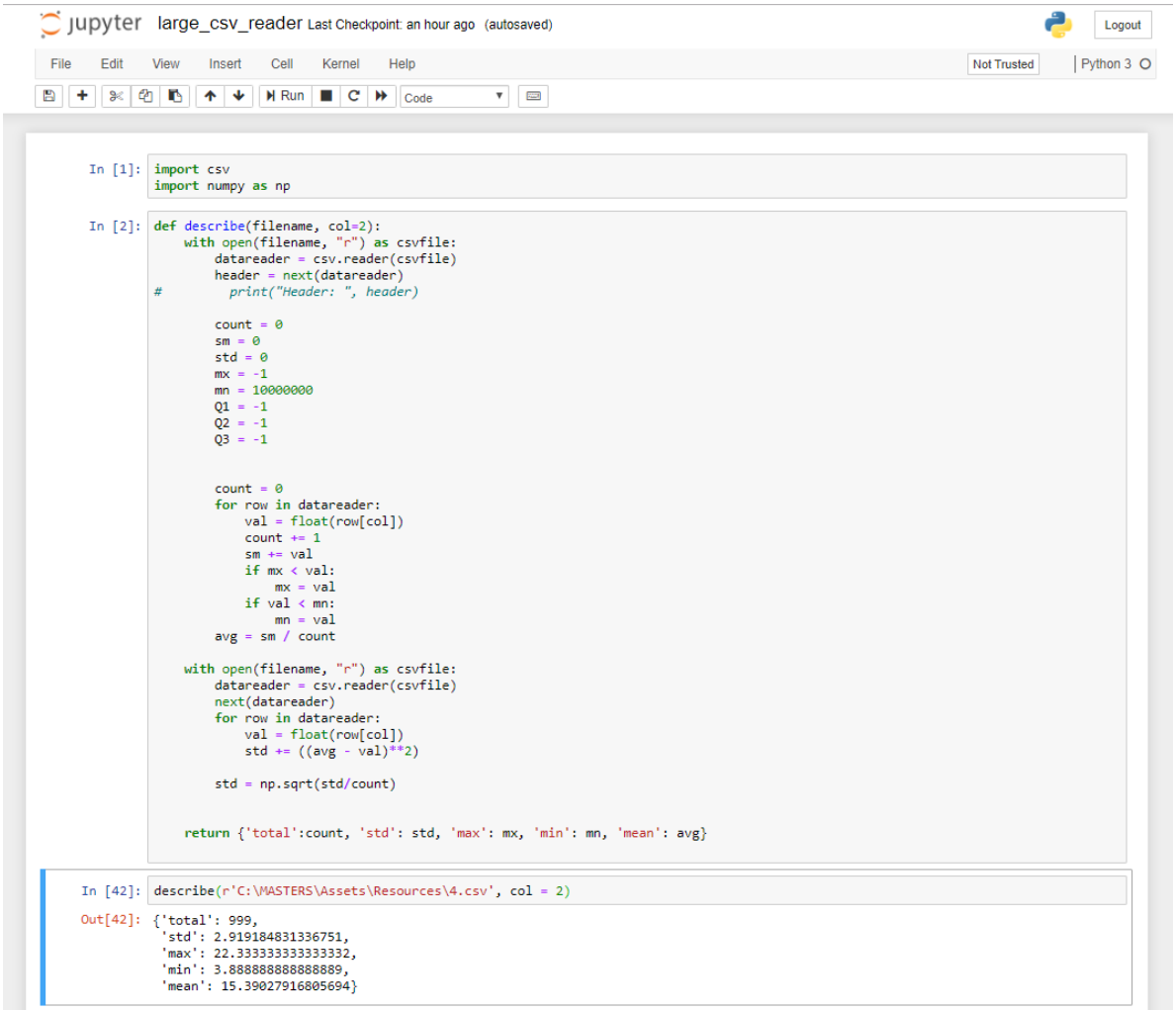

Figura 37. Execução do código para calcular estatísticas descritivas de um conjunto de dados com Jupyter Notebooks.

```
import csv
import numpy as np


def describe(filename, col=2):
    with open(filename, "r") as csvfile:
```

```
        datareader = csv.reader(csvfile)
        header = next(datareader)
#         print("Header: ", header)
        count = 0
        sm = 0
        std = 0
        mx = -1
        mn = 10000000
        Q1 = -1
        Q2 = -1
        Q3 = -1


        count = 0
        for row in datareader:
            val = float(row[col])
            count += 1
            sm += val
            if mx < val:
                mx = val
            if val < mn:
                mn = val
        avg = sm / count

    with open(filename, "r") as csvfile:
        datareader = csv.reader(csvfile)
        next(datareader)
        for row in datareader:
            val = float(row[col])
            std += ((avg - val)**2)

        std = np.sqrt(std/count)


    return {'total':count, 'std': std, 'max': mx, 'min': mn, 'mean': avg}
            :count, 'std': std, 'max': mx, 'min': mn, 'mean': avg}

describe(r'C:\MASTERS\Assets\Resources\4.csv', col = 2)
```

XXI. Inquéritos

Inquérito de visualização:

Português – https://github.com/tiago-peres/inqueritos/blob/master/Inqu%C3%A9rito%20de%20visualiza%C3%A7%C3%A3o%20-%20Google%20Forms.pdf

Inglês - https://github.com/tiago-peres/inqueritos/blob/master/English%20-%20Inqu%C3%A9rito%20de%20visualiza%C3%A7%C3%A3o%20-%20Google%20Forms.pdf

Inquérito de apreensão:

Português – https://github.com/tiago-peres/inqueritos/blob/master/Inqu%C3%A9rito%20de%20apreens%C3%A3o%20-%20Google%20Forms.pdf

Inglês - https://github.com/tiago-peres/inqueritos/blob/master/English%20-%20Inqu%C3%A9rito%20de%20apreens%C3%A3o%20-%20Google%20Forms.pdf

XXII. Resultados dos Inquéritos

Inquérito de visualização: https://github.com/tiago-peres/inqueritos/blob/master/Inqu%C3%A9rito%20de%20visualiza%C3%A7%C3%A3o.csv.zip

Inquérito de apreensão: https://github.com/tiago-peres/inqueritos/blob/master/Inqu%C3%A9rito%20de%20apreens%C3%A3o.csv.zip